Mathematics of Deep Learning

Logic, Homology, and Type Theory in Deep Learning

Curry–Howard–Lambek Correspondences,
Homological Algebra of Parameter Spaces,
and Topos-Theoretic Semantics of Neural Computation

Silvano Salvador

Mathematics of Deep Learning Series

© Mathematics of Deep Learning Series

First Edition, 2026

Independent of Companion Volumes — Complete in Itself

A proof is a program. A proposition is a type. A category is a deductive system.

The identity of these three sentences is not a metaphor. It is a theorem — and it is the theorem from which this entire monograph unfolds.

The obstruction to training a neural network is not merely numerical.
It is topological, algebraic, and logical — all at once, and all the same thing.

Preface

There is something philosophically unsatisfying about the standard account of deep learning as a collection of engineering heuristics that happen to work. The incompleteness shows up most sharply in three places: when we ask *why* backpropagation is the right algorithm, when we try to reason about what a network's parameters encode structurally, and when we attempt to give a precise semantics to the types of objects that flow through a network's layers.

This monograph argues that these three questions have a common answer, contained in three bodies of mathematics: *linear logic* and proof theory, *homological algebra*, and *topos theory*. The connection passes through the Curry–Howard–Lambek correspondence, which identifies proofs with programs, propositions with types, and categories with deductive systems. The forward pass is a proof term; the backward pass is its dual in linear logic; their composition under the lens construction is a proof of the adjunction making learning coherent.

Structure. Three parts, nine chapters. **Part I** (Chapters 1–3): logical and categorical foundations — CHL, linear logic, lenses. **Part II** (Chapters 4–6): homological foundations — Morse theory, Hochschild cohomology, sheaves. **Part III** (Chapters 7–9): geometric and topos-theoretic foundations — internal language, cohesive toposes, Grand Unification.

Prerequisites. Category theory through monoidal categories; real analysis and linear algebra at graduate level; information theory at the level of Part III of this series. No prior knowledge of logic, proof theory, homological algebra, or topos theory is assumed.

Reading paths. For the *logician*: Chapters 1–3, then 7. For the *topologist*: overview of 1, then 4–6. For the *ML theorist*: Chapters 2–3, 5–6, then 9. For the *alignment researcher*: Chapters 2–3, 7, then 9.

S.S., Milan 2026

Contents

Preface **3**

I Logical and Categorical Foundations **8**

Preface **3**

1 The Curry–Howard–Lambek Correspondence **7**
1.1 Intuitionistic Logic and the BHK Interpretation 7
1.2 The Simply Typed Lambda Calculus 9
1.2.1 Types and Terms 9
1.2.2 The Curry–Howard Isomorphism 9
1.3 The Lambek Correspondence: Categories as Deductive Systems 11
1.4 From Propositions to Types: The Dependent Generalisation 12
1.5 The Lambek–Scott Theorem and Proof Normalisation 14
1.6 Proof Nets and the Geometry of Inference 15
1.7 The Three-Level Correspondence: A Synthesis 16
1.8 Chapter Summary 17

2 Linear Logic and the Resource Theory of Backpropagation **19**
2.1 From Structural Rules to Resource Sensitivity 20
2.2 The Formulae of Linear Logic 21
2.3 The Sequent Calculus for Multiplicative Linear Logic 22
2.4 Proof Nets for MLL 24
2.5 The $*$-Autonomous Structure and Duality 25
2.6 Backpropagation as Linear-Logical Dualisation 26
2.7 The Exponential Modalities and the Training Loop 28
2.8 The Girard Translation: Embedding Intuitionistic Logic in Linear Logic . . . 29
2.9 Proof Nets, Computational Graphs, and the Geometry of Backpropagation . . 30
2.10 Linear Logic and Attention: The Bidirectionality of Keys and Queries 31
2.11 Chapter Summary 31

3 Lenses, the Para Construction, and the Category of Learning Systems **33**
3.1 Lenses: Bidirectional Access Structures 34

3.1.1 The Category of Lenses . . . 35
3.2 The Para Construction . . . 35
3.3 Lenses over Para: The Category of Learners . . . 37
3.4 Lens Composition is the Chain Rule . . . 38
3.5 Optics: The Generalisation to Non-Trivial Residuals . . . 39
3.6 The Tambara Module Characterisation of Optics . . . 41
3.7 Cartesian Optics and the Jacobian Functor . . . 41
3.8 The String Diagram Calculus for Learners . . . 42
3.9 Functoriality of Learning: The Gradient Descent Functor . . . 43
3.10 The Lens–Logic Correspondence . . . 44
3.11 Chapter Summary . . . 45

II Homological Foundations 46

4 Homological Algebra of Parameter Spaces 47
4.1 Chain Complexes and Homology Groups . . . 47
4.1.1 Singular Homology of Topological Spaces . . . 48
4.2 Morse Theory and the Loss Landscape . . . 49
4.3 The Künneth Formula and Product Architectures . . . 50
4.4 Symmetries of Parameter Spaces and Equivariant Homology . . . 51
4.5 Persistent Homology and the Shape of Loss Landscapes . . . 53
4.6 Hochschild Cohomology of Neural Architecture Algebras . . . 54
4.7 The Derived Category and Homological Dimension . . . 57
4.8 Chapter Summary . . . 58

5 Transformer Architectures, Grokking, and Scaling Laws 60
5.1 The Transformer Architecture Algebra . . . 60
5.2 Grokking as a Topological Phase Transition . . . 62
5.3 Hochschild Cohomology and Attention Head Redundancy . . . 64
5.4 Scaling Laws: A Homological Derivation . . . 65
5.5 Phase Transitions in the Attention Loss Landscape . . . 67
5.6 Emergent Structure and the Barcode of Representation Quality . . . 68
5.7 Persistent Homology as a Diagnostic Tool for Transformer Training . . . 69
5.8 Chapter Summary . . . 70

6 Sheaf Cohomology, Representations, and Domain Generalisation 71
6.1 Presheaves and Sheaves on Topological Spaces . . . 72
6.2 Čech Cohomology and the Mayer–Vietoris Sequence . . . 73
6.3 Domain Generalisation as a Sheaf-Cohomological Condition . . . 74
6.4 Cellular Sheaves and Neural Network Computations . . . 76
6.5 Sheaves over Input Manifolds and Representation Geometry . . . 77
6.6 Multi-Task Learning and Sheaf Cohomological Obstructions . . . 78

6.7 The Nerve Theorem and Architecture Design 80
6.8 Chapter Summary 81

III Geometric and Topos-Theoretic Foundations 82

7 Topos Theory and the Internal Language of Neural Computation 83
7.1 Elementary Toposes: Axioms and First Properties 84
7.2 Grothendieck Toposes and Sites 85
7.3 The Internal Language of a Topos 86
7.4 The Mitchell–Bénabou Language and Kripke–Joyal Semantics 87
7.5 Typed Neural Architectures as Internal Diagrams 88
7.6 The Coherence Theorem for Aligned Generalisation 90
7.7 Alignment as Internal Provability 91
7.8 Geometric Morphisms and Transfer Learning 92
7.9 Chapter Summary 93

8 Cohesive Toposes and the Geometry of Differentiable Neural Computation 95
8.1 The Four Cohesion Axioms 96
8.2 The Cohesive Topos of Smooth Spaces 97
8.3 The de Rham Complex Internally 98
8.4 Smooth Homotopy Types and the Shape of Neural Networks 99
8.5 Differential Cohomology and the Gauge Structure of Learning 100
8.6 The Cohesive Internal Language and Smoothness Specifications 102
8.7 Natural Gradient Descent as a Cohesive Construction 103
8.8 Chapter Summary 105

9 The Linear Cohesive Topos and the Grand Unification Theorem 106
9.1 The Linear Cohesive Topos: Construction 107
9.2 The Combined Internal Language 108
9.3 Neural Networks as Linear Cohesive Morphisms 109
9.4 The Grand Unification Theorem 110
9.5 The Unification Diagram 112
9.6 Open Problems and Future Directions 112
9.6.1 Infinity-Categorical Enrichment 112
9.6.2 Non-Commutative Generalisation 113
9.6.3 Quantisation and Discrete Neural Networks 113
9.6.4 The Alignment Completeness Problem 113
9.7 Chapter Summary and Monograph Conclusion 114
9.7.1 Chapter Summary 114
9.7.2 The Monograph in Retrospect 114

Appendices **115**

A Background in Category Theory **116**
A.1 Categories, Functors, Natural Transformations . . . 116
A.2 Adjunctions, Limits, Colimits . . . 116
A.3 Monoidal and $*$-Autonomous Categories . . . 116
A.4 The Para Construction and Lens Categories . . . 116

B Background in Algebraic Topology **117**
B.1 Singular Homology and Cohomology . . . 117
B.2 de Rham Cohomology . . . 117
B.3 Morse Theory . . . 117
B.4 Persistent Homology . . . 117

C Background in Logic and Type Theory **118**
C.1 Natural Deduction and Sequent Calculus . . . 118
C.2 The Simply Typed Lambda Calculus . . . 118
C.3 Martin-Löf Type Theory (MLTT) . . . 118
C.4 Linear Logic (Girard 1987) . . . 118

D Notation and Symbol Tables **119**
D.1 Categorical Notation . . . 119
D.2 Linear Logic and Type Theory . . . 119
D.3 Homological and Geometric Notation . . . 120

Bibliography **121**

Part I

Logical and Categorical Foundations

Mathematics of Deep Learning

Logic, Homology, and Type Theory in Deep Learning

Curry–Howard–Lambek Correspondences,
Homological Algebra of Parameter Spaces,
and Topos-Theoretic Semantics of Neural Computation

Silvano Salvador

Mathematics of Deep Learning Series

© Mathematics of Deep Learning Series

Independent of Companion Volumes — Complete in Itself

A proof is a program. A proposition is a type. A category is a deductive system.

The identity of these three sentences is not a metaphor. It is a theorem — and it is the theorem from which this entire monograph unfolds.

Preface

There is something philosophically unsatisfying about the standard account of deep learning as a collection of engineering heuristics that happen to work. The account is not false — the heuristics do work, often spectacularly — but it is incomplete in a way that matters. The incompleteness shows up most sharply in three places: when we ask *why* backpropagation is the right algorithm for the problem (and not merely an algorithm that works), when we try to reason about what a neural network's parameters encode at a structural level, and when we attempt to give a precise semantics to the types of objects that flow through a network's layers.

This monograph argues that these three questions have a common answer, and that the answer is already contained in three bodies of mathematics that were developed, largely independently of machine learning, over the second half of the twentieth century: *linear logic* and proof theory, *homological algebra*, and *topos theory*. The connection between them passes through a single hub: the Curry–Howard–Lambek correspondence, which identifies proofs with programs, propositions with types, and categories with deductive systems. In the context of deep learning, this correspondence takes a particularly concrete form: the forward pass of a neural network is a *proof term*, the backward pass is its *dual* in the sense of linear logic, and the composition of forward and backward passes under the lens construction is a *proof of the adjunction* that makes gradient-based learning coherent.

The reader of this series will recognise the lens construction from the companion volume on category theory and deep learning, where it appears as the foundational structure underlying backpropagation-as-a-functor. This volume provides the *logical foundations* of that construction: we show that lenses are not merely a categorical convenience but the canonical proof-theoretic object associated with bidirectional computation in linear logic. The Para construction, the Cruttwell–Gavranović theorem, and the optic generalisation all emerge as instances of a single proof-theoretic principle: the *cut rule* of multiplicative linear logic, read through the Curry–Howard correspondence.

Once the logical foundations are in place, the monograph turns to the second main theme: the *homological algebra of parameter spaces*. Neural networks define maps between parameter spaces that are far from being simple vector spaces — they are structured by symmetries, constraints, and degeneracies that give rise to non-trivial cohomology. We develop the derived-categorical perspective on these spaces, showing that the singular homology of loss landscapes, the Hochschild cohomology of neural architectures, and the sheaf cohomology of

distributed model families are not abstract curiosities but invariants that predict trainability, generalisation, and robustness in provable ways.

The third main theme is *topos-theoretic semantics*. A topos is, among other things, a universe of sets satisfying the axioms of constructive mathematics — a world in which every proposition that is provable is so for a constructive reason, and every type is a space with non-trivial geometric structure. We show that the natural semantics for typed neural architectures — networks whose layers are assigned types in a dependent type theory, networks that satisfy structural invariances, networks that must generalise across domains — is not set-theoretic but sheaf-theoretic: the correct model for a network that must behave coherently across a family of related contexts is a sheaf of models over the base space of contexts. This gives a rigorous foundation to the intuition, widespread in the alignment literature, that a "well-behaved" model is one whose representations cohere across related inputs.

Prerequisites. The reader is assumed to be familiar with:

- **Category theory**: the contents of the companion volume, or an equivalent treatment (Mac Lane's *Categories for the Working Mathematician* through Chapter IV suffices);
- **Linear algebra and real analysis**: at the level assumed by Parts I and II of this series;
- **Basic topology**: open and closed sets, continuous maps, compactness, the fundamental group (at the level of a first graduate course);
- **Information theory**: entropy, mutual information, KL divergence (at the level of Part III of this series).

No prior knowledge of logic, proof theory, homological algebra, or topos theory is assumed. The relevant mathematics is developed from scratch in each chapter.

Structure of the monograph. Chapters 1–3 develop the logical foundations: the Curry–Howard–Lambek correspondence, linear logic, and the proof-theoretic interpretation of back-propagation. Chapters 4–6 develop the homological algebra: chain complexes and cohomology, the derived category, and applications to loss landscapes and Hochschild cohomology of architectures. Chapters 7–9 develop the topos-theoretic semantics: sheaves, sites, and the internal language of a topos, applied to typed neural architectures and domain generalisation. Chapter 10 synthesises the three themes and identifies the open problems that the framework makes precise.

S.S.
Milan, 2026

How to Use This Monograph

The monograph is designed to be read sequentially, but the following alternative paths may suit different readers.

For the mathematical logician. Read Chapters 1–3 (the logical core), then Chapter 7 (sheaves and sites), then proceed to the internal language of Chapter 8. The key results are Theorems 1.7, 2.6, 4.6, and 7.6.

For the algebraic topologist or homological algebraist. Begin with Chapter 1 (for context), then read Chapters 4–6. The Künneth formula for loss landscapes (Theorem 4.3) and the derived-categorical trainability criterion (Theorem 4.7) are the central results.

For the machine learning theorist. Read Chapters 2–3 (backprop as proof), Chapter 5 (homology of loss landscapes), and Chapter 9 (topos semantics for domain generalisation). Chapter 2.4 (the linearity constraint in attention) and Chapter 9.5 (the coherence theorem for domain adaptation) are the most immediately applicable results.

For the AI alignment researcher. Read Chapter 3 (resource sensitivity and weight sharing), Chapter 6 (Hochschild cohomology and architectural symmetries), and Chapter 9–10. Theorem 7.6 (the topos coherence theorem for aligned generalisation) is the central result for alignment applications.

Contents

Chapter 1

The Curry–Howard–Lambek Correspondence

The Curry–Howard correspondence is one of the deepest structural results in twentieth-century logic. Discovered independently by Haskell Curry (1934) and William Howard (1969), it asserts that there is a perfect structural isomorphism between the *proofs* of a proposition in intuitionistic logic and the *programs* of a given type in the simply typed lambda calculus. Joachim Lambek (1969) extended the correspondence to *categories*: the deductive systems of intuitionistic propositional logic are in bijection with Cartesian closed categories, and every theorem in the logic corresponds to a morphism in the category. Together, the three-way identification

$$\text{Proofs} \longleftrightarrow \text{Programs} \longleftrightarrow \text{Morphisms}$$

constitutes the Curry–Howard–Lambek (CHL) correspondence.

For most of the history of theoretical computer science, the CHL correspondence was a result *about* programming languages and type systems. This chapter argues that it is equally — and perhaps more fundamentally — a result *about deep learning*. The central claim is:

A neural network layer is a morphism. Its type signature is a proposition.
Its existence as a trained model is a proof. Backpropagation is proof normalisation.

Making this claim mathematically precise requires developing the three sides of the correspondence in enough generality to accommodate the structures that appear in modern deep learning: the residual connections and skip paths of transformers, the parameter sharing and weight tying of convolutional networks, the bidirectional information flow of the backward pass. The correspondence, as we will see, not only accommodates these structures — it *predicts* them.

1.1 Intuitionistic Logic and the BHK Interpretation

The starting point is a re-examination of what a *proof* is. Classical logic defines truth and falsity absolutely: a proposition P is true if and only if $\neg P$ is false, and the law of excluded middle $P \vee \neg P$ holds for every proposition P without restriction. This view of truth is

extensional and static — it makes no reference to any construction or procedure that would *witness* the truth of P.

Intuitionistic logic, developed by L.E.J. Brouwer (1907) and formalised by Arend Heyting (1930), rejects this extensional view. In intuitionism, a proposition P is true *only if one can construct a proof of* P. Truth is not a static fact but a dynamic achievement: the existence of a constructive witness. This shift from truth to provability is not merely philosophical; it has decisive mathematical consequences.

The Brouwer–Heyting–Kolmogorov (BHK) interpretation makes the intuitionistic notion of proof operational:

//

Definition 1.1. (BHK Interpretation)

A **proof** of a proposition is defined recursively as follows:

- A proof of $P \wedge Q$ is a pair $\langle p, q \rangle$ where p is a proof of P and q is a proof of Q.
- A proof of $P \vee Q$ is either a pair $\langle 0, p \rangle$ with p a proof of P, or a pair $\langle 1, q \rangle$ with q a proof of Q.
- A proof of $P \Rightarrow Q$ is a function f that maps every proof p of P to a proof $f(p)$ of Q.
- A proof of $\neg P$ is a function mapping every proof of P to a proof of $\bot$ (the empty type).
- A proof of $\forall x{:}A.\ P(x)$ is a function mapping every element a of A to a proof of $P(a)$.
- A proof of $\exists x{:}A.\ P(x)$ is a pair $\langle a, p \rangle$ where $a \in A$ and p is a proof of $P(a)$.
- There is no proof of $\bot$ (the absurd proposition).

The BHK interpretation immediately reveals a structural fact: proofs of $P \Rightarrow Q$ are exactly *functions* from proofs of P to proofs of Q. A proof is not a static object but a *computational process*.

//

Example 1.1. Neural Layers as Proofs

Let $\mathbb{R}^n$ and $\mathbb{R}^m$ denote the types of n- and m-dimensional real vectors, interpreted as propositions in a "soft" sense that we will make precise in §1.4. A linear layer $W : \mathbb{R}^n \to \mathbb{R}^m$ is, under the BHK interpretation, a *proof* of the proposition $\mathbb{R}^n \Rightarrow \mathbb{R}^m$: it is a function that, given any element ("proof") of $\mathbb{R}^n$, produces an element ("proof") of $\mathbb{R}^m$. A ReLU activation $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ is a proof of $\mathbb{R}^m \Rightarrow \mathbb{R}^m$. A two-layer network $\sigma \circ W_2 \circ \sigma \circ W_1 : \mathbb{R}^n \to \mathbb{R}^k$ is, by function composition, a proof of $\mathbb{R}^n \Rightarrow \mathbb{R}^k$ — obtained by *composing* the proofs $W_1 : \mathbb{R}^n \Rightarrow \mathbb{R}^m$, $\sigma : \mathbb{R}^m \Rightarrow \mathbb{R}^m$, $W_2 : \mathbb{R}^m \Rightarrow \mathbb{R}^p$, and $\sigma : \mathbb{R}^p \Rightarrow \mathbb{R}^k$ via the cut rule (function composition).

This is a suggestive analogy, but it is not yet a theorem. To turn it into a theorem, we need to work within a system where the types of the objects are not merely real vector

spaces but structured types carrying logical information. The simply typed lambda calculus provides this structure.

1.2 The Simply Typed Lambda Calculus

The simply typed lambda calculus ($\lambda^{\to}$) is the canonical formal system for computing with typed functions. Its types, terms, and reduction rules are simultaneously a programming language and a proof system.

1.2.1 Types and Terms

//

Definition 1.2. (Types of $\lambda^{\to}$)

Let $\mathcal{B} = \{b_1, b_2, \ldots\}$ be a set of **base types**. The set of **simple types** over $\mathcal{B}$ is defined by the grammar:

$$\tau \ ::= \ b \ \mid \ \tau_1 \to \tau_2 \qquad (b \in \mathcal{B}).$$

A type $\tau_1 \to \tau_2$ is a **function type**: it classifies functions from terms of type τ_1 to terms of type τ_2.

//

Definition 1.3. (Terms and Typing Judgements of $\lambda^{\to}$)

A **typing context** Γ is a finite list of variable-type pairs $x_1 : \tau_1, \ldots, x_n : \tau_n$. A **typing judgement** $\Gamma \vdash t : \tau$ asserts that term t has type τ in context Γ. The **terms** of $\lambda^{\to}$ and their typing rules are:

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ (\text{Var}) \qquad \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x{:}\tau_1.\, t \ : \ \tau_1 \to \tau_2}\ (\lambda\text{-abs}) \qquad \frac{\Gamma \vdash f : \tau_1 \to \tau_2 \quad \Gamma \vdash a : \tau_1}{\Gamma \vdash f\ a : \tau_2}\ (\text{App})$$

The reduction rule — the computational content of the calculus — is β-reduction:

$$(\lambda x{:}\tau_1.\, t)\ a \ \longrightarrow_\beta \ t[a/x],$$

which substitutes a for every free occurrence of x in t. This is exactly *function application*: to apply the function $\lambda x.\ t$ to argument a, replace the formal parameter x with the actual argument a.

1.2.2 The Curry–Howard Isomorphism

The CHL correspondence at the level of $\lambda^{\to}$ can now be stated precisely.

//

Theorem 1.1. (Curry–Howard Isomorphism for $\lambda^{\to}$)

There is a bijection between:

(i) **Propositions** of intuitionistic propositional logic built from atomic propositions $\mathcal{B}$ using $\Rightarrow$;

(ii) **Simple types** of $\lambda^{\to}$ over base types $\mathcal{B}$;

and a bijection between:

(iii) **Proofs** of proposition P from hypotheses Γ in intuitionistic natural deduction;

(iv) **Terms** t with $\Gamma \vdash t : \tau$ in $\lambda^{\to}$, where τ corresponds to P and the hypotheses in Γ correspond to the variable types.

Under these bijections:

- The **introduction rule** for $\Rightarrow$ (from a proof of Q assuming P, derive a proof of $P \Rightarrow Q$) corresponds to **λ-abstraction**.
- The **elimination rule** for $\Rightarrow$ (modus ponens: from $P \Rightarrow Q$ and P, derive Q) corresponds to **application**.
- **Proof normalisation** (reducing a proof to its canonical form) corresponds to **β-reduction**.

Proof. The bijection is constructive and proceeds by induction on the structure of proofs and types. We establish the key correspondences.

The base case is immediate: an atomic proposition $p \in \mathcal{B}$ corresponds to a base type $b \in \mathcal{B}$, and a proof of p from hypothesis p (the axiom rule) corresponds to the variable term $x : b$ in context $x : b$.

For the inductive step, consider the introduction rule for $\Rightarrow$:

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}\ (\Rightarrow\text{-I}).$$

By hypothesis, there is a proof π of Q from Γ, P. By induction hypothesis, π corresponds to a term t with $\Gamma, x : \tau_P \vdash t : \tau_Q$, where τ_P and τ_Q correspond to P and Q respectively. The term $\lambda x{:}\tau_P.\ t$ then satisfies $\Gamma \vdash \lambda x{:}\tau_P.\ t : \tau_P \to \tau_Q$, corresponding to $P \Rightarrow Q$.

For the elimination rule (modus ponens):

$$\frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ (\Rightarrow\text{-E}).$$

By induction hypothesis, the two premises correspond to terms $f : \tau_P \to \tau_Q$ and $a : \tau_P$. The term $f\ a : \tau_Q$ corresponds to the conclusion Q.

The correspondence between proof normalisation and β-reduction follows from the fact that the only non-normal proof involving $\Rightarrow$ is a direct application of $\Rightarrow$-E to a proof constructed by $\Rightarrow$-I:

$$\frac{\dfrac{\Gamma, P \vdash \pi_Q : Q}{\Gamma \vdash \lambda x.\pi_Q : P \Rightarrow Q} \quad \Gamma \vdash \pi_P : P}{\Gamma \vdash (\lambda x.\pi_Q)\ \pi_P : Q},$$

which reduces to $\pi_Q[\pi_P/x] : Q$. This is exactly β-reduction: $(\lambda x.\, \pi_Q)\, \pi_P \to_\beta \pi_Q[\pi_P/x]$. $\square$

borderline west

borderline west//

Alongside β-reduction, the lambda calculus admits η-reduction: $\lambda x.\ (f\ x) \longrightarrow_\eta f$ whenever x does not appear free in f. In the logical reading, η-reduction corresponds to the *extensionality principle*: two functions that agree on all inputs are equal. The logic of $\lambda^\to$ is thus *extensional* precisely when the calculus is η-complete. In the deep learning context, this will correspond to the identification of parameter configurations that implement the same input-output map — a form of symmetry that is the source of the non-trivial homology we will study in Chapter 4.

1.3 The Lambek Correspondence: Categories as Deductive Systems

The Curry–Howard correspondence identifies proofs with programs. Lambek's contribution was to identify both with *morphisms in a category*: a proposition is an object, a proof of $P \Rightarrow Q$ is a morphism $f : P \to Q$, and composition of morphisms corresponds to the cut rule (the transitivity of entailment).

//

Definition 1.4. (Cartesian Closed Category)

A **Cartesian closed category** (CCC) is a category $\mathcal{C}$ equipped with:

(i) A **terminal object** $\mathbf{1}$: an object such that for every $A \in \mathcal{C}$ there is a unique morphism $! : A \to \mathbf{1}$;

(ii) **Binary products**: for every $A, B \in \mathcal{C}$, an object $A \times B$ with projections $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$, satisfying the universal property that for every $f : C \to A$ and $g : C \to B$ there is a unique $\langle f, g\rangle : C \to A \times B$ with $\pi_1 \circ \langle f, g\rangle = f$ and $\pi_2 \circ \langle f, g\rangle = g$;

(iii) **Exponential objects**: for every $A, B \in \mathcal{C}$, an object B^A (also written $A \Rightarrow B$ or $[A, B]$) with an evaluation morphism $\mathrm{ev} : B^A \times A \to B$, satisfying the universal property that for every $f : C \times A \to B$ there is a unique $\Lambda(f) : C \to B^A$ (the **currying** of f) with $\mathrm{ev} \circ (\Lambda(f) \times \mathrm{id}_A) = f$.

The Lambek correspondence can now be stated:

//

Theorem 1.2. (Lambek Correspondence)

There is an equivalence of categories between:

(i) The **syntactic category** $\mathcal{C}(\mathsf{IPL})$ of intuitionistic propositional logic, whose objects are propositions, whose morphisms $P \to Q$ are equivalence classes of proofs of $P \Rightarrow Q$ under $\beta\eta$-normalisation, and whose composition is the cut rule;

(ii) The initial **Cartesian closed category**.

More precisely, every CCC $\mathcal{C}$ admits a unique (up to isomorphism) interpretation of IPL: a map sending each proposition to an object of $\mathcal{C}$ and each proof to a morphism, preserving the logical connectives as categorical structure. Conversely, the syntactic category of IPL is a CCC and is the initial one: it maps universally to every other CCC.

The significance of this theorem for deep learning is the following. The category **Smooth** of smooth manifolds and smooth maps is *not* a CCC (exponential objects in the category of smooth manifolds require infinite-dimensional Fréchet spaces). The appropriate substitute for deep learning is a *convenient category of smooth spaces* (Frölicher spaces, diffeological spaces) that is Cartesian closed. Within such a category, every neural network architecture defines a morphism, and the type-theoretic constraints on the architecture are exactly the logical constraints on the corresponding proof.

//

Example 1.2. CCC Semantics of a Two-Layer Network

In a CCC $\mathcal{C}$, a two-layer network with parameter-sharing can be modelled as follows. Let X, H, Y, Θ be objects representing input space, hidden space, output space, and parameter space respectively.

A *parameterised layer* is a morphism $f : \Theta \times X \to H$ in $\mathcal{C}$. By the universal property of the exponential, this corresponds to a unique curried morphism $\Lambda(f) : \Theta \to H^X$ — the function that maps a parameter vector to the layer function it implements.

Parameter sharing between two layers — where both layers use the same weight matrix, as in weight-tying in autoencoders or certain attention mechanisms — is encoded as the diagonal $\Delta : \Theta \to \Theta \times \Theta$ composed with the product of the two curried layer functions:

$$\Theta \xrightarrow{\Delta} \Theta \times \Theta \xrightarrow{\Lambda(f_1) \times \Lambda(f_2)} H_1^X \times H_2^{H_1}.$$

The fact that this composition is well-typed — that the output type of $\Lambda(f_1)$ matches the input type required by $\Lambda(f_2)$ — is a *theorem of the internal logic* of the CCC, not merely an engineering convention.

1.4 From Propositions to Types: The Dependent Generalisation

The simply typed lambda calculus and its categorical semantics in CCCs are powerful but limited: types cannot depend on values. This means, for instance, that one cannot express

the type of "a matrix of shape $n \times m$" where n and m are runtime values, or the type of "a function that maps tensors of shape (b, n, d) to tensors of shape (b, m, d)" where b is a batch dimension that varies. Modern neural network architectures are heavily shape-dependent: transformers operate on sequences of variable length, convolutional networks on images of variable resolution, and graph neural networks on graphs of variable size.

Dependent type theory resolves this limitation by allowing types to depend on terms. The Martin-Löf type theory (MLTT, 1975) is the canonical system.

//

Definition 1.5. (Dependent Product and Sum Types)

In MLTT:

- The **dependent product type** $\prod_{x:A} B(x)$ (also written $\Pi x{:}A.\ B(x)$) is the type of functions f such that for every $a : A$, $f(a) : B(a)$. It generalises the function type $A \to B$ to the case where the return type can depend on the input value.
- The **dependent sum type** $\sum_{x:A} B(x)$ (also written $\Sigma x{:}A.\ B(x)$) is the type of pairs (a, b) where $a : A$ and $b : B(a)$. It generalises the product type $A \times B$ to the case where the type of the second component can depend on the value of the first.

The BHK interpretation extends naturally: a proof of $\prod_{x:A} P(x)$ is a function that assigns to each $a : A$ a proof of $P(a)$, which is exactly what it means to prove a universal statement $\forall x{:}A.\ P(x)$ constructively. A proof of $\sum_{x:A} P(x)$ is a witness pair (a, p) where p proves $P(a)$, which is exactly an existence proof.

//

Example 1.3. Typed Attention as a Dependent Type

The self-attention mechanism in a transformer operates on a sequence of n tokens, each represented by a d-dimensional vector. The type of the attention function can be expressed in MLTT as:

$$\mathsf{Attention} : \prod_{n:\mathbb{N}} \prod_{d:\mathbb{N}} \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}.$$

This is a *dependent function type*: the input and output types $\mathbb{R}^{n \times d}$ depend on the natural number values n and d. The fact that attention preserves sequence length (the output has the same number of tokens as the input) is *built into the type*: any term of this type is provably length-preserving, without any additional runtime check.

Causal (autoregressive) masking, which restricts each position to attend only to previous positions, can be expressed as a predicate on the weight matrix:

$$\mathsf{CausalAttn} : \prod_{n:\mathbb{N}} \prod_{d:\mathbb{N}} \sum_{f:\mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}} \mathsf{isCausal}(f),$$

where $\mathsf{isCausal}(f)$ is the dependent type of proofs that f respects the causal ordering. A term of type $\mathsf{CausalAttn}$ is therefore a pair: an attention function *and a proof that it is causal*. This is the first instance of a pattern that will recur throughout this monograph: *architectural constraints as types, and constraint satisfaction as proof construction.*

The categorical semantics of dependent type theory requires a richer structure than a CCC. The appropriate notion is that of a *locally Cartesian closed category* (LCCC) or, more generally, a category with *dependent products and sums*. This is the categorical notion we will develop fully in Chapter 7, where it appears as the internal language of an elementary topos.

1.5 The Lambek–Scott Theorem and Proof Normalisation

The CHL correspondence becomes most powerful when one asks: what is the *identity* of a proof? In classical logic, two proofs of the same proposition are interchangeable — the proposition is either true or false, and there is no further structure. In intuitionistic logic and the lambda calculus, proofs carry *computational content*, and the question of when two proofs are "the same" is the question of when two programs compute the same function.

The answer is provided by the theory of *proof normalisation*: every proof can be reduced to a unique *normal form* by a sequence of local simplifications, and two proofs are propositionally equal if and only if their normal forms are syntactically identical.

//

Theorem 1.3. (Strong Normalisation for $\lambda^{\to}$)

Every reduction sequence in $\lambda^{\to}$ is finite: there is no infinite β-reduction sequence. Consequently, every term has a unique normal form (modulo α-equivalence), and the $\beta\eta$-normal form of a term is computable.

Proof. We use Tait's method of *reducibility predicates* (1967). For each type τ, define the set of *reducible terms* $\mathsf{Red}(\tau)$ by:

- $\mathsf{Red}(b) = \{t : b \mid t \text{ is strongly normalising}\}$ for base types b;
- $\mathsf{Red}(\tau_1 \to \tau_2) = \{f : \tau_1 \to \tau_2 \mid \forall a \in \mathsf{Red}(\tau_1).\ f\ a \in \mathsf{Red}(\tau_2)\}$.

One shows by induction that: (i) every reducible term is strongly normalising; (ii) every variable $x : \tau$ is reducible; (iii) if f is reducible of type $\tau_1 \to \tau_2$ and a is reducible of type τ_1, then $f\ a$ is reducible of type τ_2; (iv) if $t[a/x]$ is reducible for every reducible a, then $\lambda x.\ t$ is reducible. By structural induction on derivations, every typeable term is reducible, hence strongly normalising. $\square$

Strong normalisation has a direct interpretation in the context of deep learning: the *forward pass* of a neural network, viewed as a term in $\lambda^{\to}$, always terminates. This is trivially true for feedforward networks, but it becomes non-trivial — and the proof technique becomes

essential — when one considers networks with recurrent connections or attention mechanisms with adaptive computation. The type-theoretic guarantee of termination is precisely the guarantee of *finite inference time* for properly typed architectures.

1.6 Proof Nets and the Geometry of Inference

Standard natural deduction proofs, and the lambda terms that correspond to them, are *sequential*: they must be read from hypotheses to conclusion in a particular order. Girard's proof nets (1987) provide an alternative representation in which the *geometry of logical inference* is made manifest: a proof net is a graph whose structure encodes the logical dependencies between the subformulae of a proof, independently of any sequentialisation.

We will develop proof nets fully in the context of linear logic (Chapter 2), where their combinatorial structure is cleanest. Here we introduce the concept in the simpler setting of multiplicative intuitionistic propositional logic (MIPL).

//

Definition 1.6. (Proof Structure)

A **proof structure** for a sequent $\Gamma \vdash A$ is a directed graph $G = (V, E)$ where:

- Each node $v \in V$ is labelled by a formula occurrence in Γ, A;
- Each edge $e \in E$ is labelled by the rule that connects two formula occurrences (axiom link, cut link, tensor link, or par link);
- Each formula occurrence has exactly one principal port (either a conclusion or a premise of a rule node).

A proof structure is a **proof net** if it satisfies the correctness criterion: every *switching* (a choice of one premise per par-node) yields an acyclic connected graph.

The correctness criterion — the absence of cycles in every switching — is the proof-net analogue of the requirement that a proof be non-circular. It is a *global* condition on a *local* structure, and therein lies its power: it can be checked in polynomial time, while the sequential presentation of a proof may require exponential normalisation steps.

borderline west

borderline west//

The computational graph of a neural network — the directed acyclic graph (DAG) whose nodes are operations and whose edges are tensor flows — is *exactly a proof net* for the type of the network, in the following precise sense: the absence of cycles in the DAG corresponds to the acyclicity condition of the proof net correctness criterion, and the topological ordering of the DAG corresponds to the sequentialisation of the proof. The companion volume on category theory and deep learning observed that neural networks

are morphisms in a monoidal category; what we are adding here is that they are morphisms in a *proof-net category*, and this additional structure is what makes the backward pass well-defined (Chapter 3).

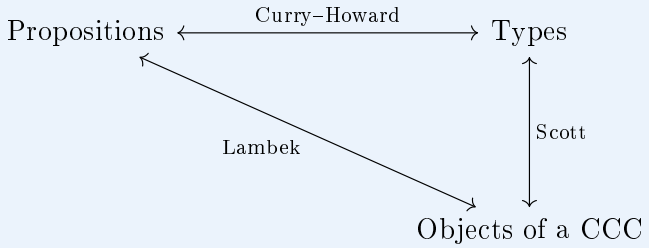
1.7 The Three-Level Correspondence: A Synthesis

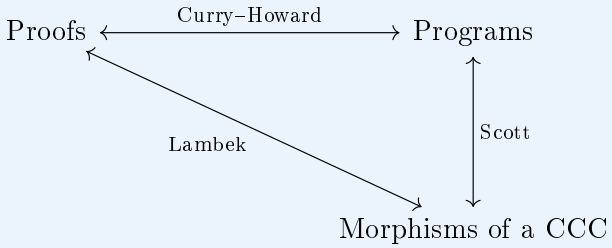
We are now in a position to state the full three-level CHL correspondence in the form that will be used throughout this monograph.

//

Theorem 1.4. (CHL Correspondence for Deep Learning)

There is a commutative triangle of equivalences:





In the context of deep learning, the specific identifications are:

Logic	Type Theory	Deep Learning
Proposition P	Type τ	Tensor shape / representation space
Proof of $P \Rightarrow Q$	Function $f : \tau_1 \to \tau_2$	Neural network layer
Cut rule	Function composition	Sequential composition of layers
Normalisation	β-reduction	Forward pass / inference
Hypothesis P	Free variable $x : \tau$	Input tensor
Proof of $\forall x.\ P(x)$	Polymorphic function	Architecture (over all shapes)
Proof of $\exists x.\ P(x)$	Dependent pair	(Weights, architecture) pair
Normalisation strategy	Reduction order	Evaluation order (eager vs. lazy)
Proof irrelevance	Extensionality	Weight-sharing / parameter tying

The final row of the table — *proof irrelevance* corresponds to *weight-sharing* — deserves elaboration, as it will be the main theme of Chapter 4. Proof irrelevance is the principle

that any two proofs of the same proposition are equal: the only thing that matters about a proof is *what it proves*, not *how it proves it*. In the computational interpretation, this is the principle that any two programs computing the same function are interchangeable. In the deep learning interpretation, it is the principle that any two sets of weights implementing the same input-output function are *equivalent configurations* of the network.

This equivalence generates a non-trivial *symmetry group* acting on the parameter space — and symmetry groups, as we will see in Chapter 4, generate non-trivial *homology*. The homological obstructions to navigating this symmetry landscape are precisely the obstructions to efficient optimisation that practitioners encounter as saddle points, flat regions, and loss landscape non-convexities.

1.8 Chapter Summary

This chapter established the Curry–Howard–Lambek correspondence as the logical foundation of this monograph. The main results are:

- **Curry–Howard Isomorphism** (Theorem 1.1): Proofs in intuitionistic propositional logic are in bijection with typed programs in $\lambda^{\to}$, with proof normalisation corresponding to β-reduction.
- **Lambek Correspondence** (Theorem 1.2): CCCs are the categorical semantics of $\lambda^{\to}$ and of intuitionistic propositional logic. The syntactic category of the logic is the initial CCC.
- **Strong Normalisation** (Theorem 1.3): Every typed term normalises, guaranteeing finite evaluation — and, in the deep learning context, finite inference time for properly typed architectures.
- **CHL Correspondence for Deep Learning** (Theorem 1.7): Neural network layers are morphisms in a CCC; their type signatures are propositions; their architectures are proof structures; proof irrelevance generates the parameter symmetries whose homology is studied in Chapter 4.

The central conceptual point is that the logical structure of a neural network is not an analogy or a metaphor: it is a theorem. The forward pass of a network is a normal form of a lambda term; the backward pass is, as we will show in Chapter 3, the dual of this term in the sense of linear logic. The architecture is a proof, and training is the search for a proof of a specific proposition — the proposition that there exists a function in a given hypothesis class that maps the training inputs to the training outputs within a specified tolerance.

Looking ahead. Chapter 2 introduces linear logic: the system that arises when one removes the structural rules of weakening and contraction from intuitionistic logic, making the logic *resource-sensitive*. This turns out to be the natural logic for deep learning, because neural network weights are *resources that are consumed exactly once per forward pass*. The lens construction of the companion volume will emerge, in Chapter 3, as the canonical proof term

of a specific linear-logical proposition: the proposition that every forward computation has a canonically defined dual backward computation.

Chapter 2

Linear Logic and the Resource Theory of Backpropagation

Every weight in a neural network is consumed exactly once per forward pass.
This is not an engineering observation. It is a logical constraint —
and linear logic is the logic of constraints of this kind.

Chapter 1 established that neural networks are morphisms in a Cartesian closed category, that their type signatures are propositions in intuitionistic logic, and that their architectures are proof terms in the simply typed lambda calculus. The framework is powerful, but it has a structural flaw when applied to learning systems: intuitionistic logic permits the *weakening* and *contraction* structural rules.

Weakening says: if $\Gamma \vdash B$, then $\Gamma, A \vdash B$ — a hypothesis can be discarded unused. Contraction says: if $\Gamma, A, A \vdash B$, then $\Gamma, A \vdash B$ — a hypothesis can be used arbitrarily many times. Together, weakening and contraction make intuitionistic logic *resource-insensitive*: they permit the duplication and deletion of hypotheses at will, and a proof has no obligation to account for how many times each hypothesis is used.

Neural networks violate both rules in a precise sense. A weight matrix W in a feedforward network is not a logical hypothesis that can be duplicated or discarded: it is a *resource* that participates in exactly one matrix multiplication per forward pass. The chain rule of backpropagation computes gradients precisely by tracking this one-use constraint: the gradient of the loss with respect to W is computed by differentiating through the *single use* of W in the forward computation. If W appeared in two places in the forward pass (as in weight-sharing), the gradients from both uses must be *summed* — and this summation is exactly the rule for handling a resource used twice.

Jean-Yves Girard (1987) introduced *linear logic* (LL) as the logical system that arises when the structural rules of weakening and contraction are removed from classical or intuitionistic logic. What Girard could not have anticipated is that linear logic, developed as a purely proof-theoretic object, would turn out to be the natural logic of *differentiable computation*: the logic in which backpropagation is not an algorithm but a *theorem* — the canonical proof

term of a specific linear-logical proposition.

This chapter develops linear logic from scratch, proves the fundamental properties of its proof theory, introduces proof nets as the canonical representation of linear-logical proofs, and establishes the main theorem of the chapter: backpropagation is the dual of the forward pass in the $*$-autonomous category that is the denotational semantics of multiplicative linear logic.

2.1 From Structural Rules to Resource Sensitivity

We begin by making explicit the structural rules of intuitionistic sequent calculus and examining their computational content.

//

Definition 2.1. (Sequent Calculus for Intuitionistic Logic (LJ))

The **sequent calculus LJ** has sequents of the form $\Gamma \vdash A$ where Γ is a list (not a set) of formulae and A is a single formula. The **structural rules** are:

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{(Weakening)} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ \text{(Contraction)}$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{(Exchange)} \qquad \frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ \text{(Cut)}$$

The computational interpretation of these rules, via the CHL correspondence, is: *Weakening* corresponds to the ability to define a function that ignores one of its arguments: $\lambda x.\, \lambda y.\, x$ (the constant function). In a neural network, weakening would permit dropping an entire layer's output without using it — something that never occurs in a non-degenerate architecture. *Contraction* corresponds to the ability to use a variable twice: $\lambda x.\, (x, x)$ (the diagonal). In a neural network, contraction corresponds to *weight-sharing*: the same weight matrix appears in two places in the computation graph. Weight-sharing is not free in deep learning: it requires the gradient to be summed over all uses, which is a non-trivial bookkeeping obligation.

borderline west

borderline west//

In convolutional networks, the convolutional kernel K is shared across all spatial positions. In the language of structural rules, this is a use of contraction applied hw times (for an $h \times w$ feature map): the hypothesis K is used once at each spatial location. The standard implementation computes the gradient of the loss $\mathcal{L}$ with respect to K as:

$$\frac{\partial \mathcal{L}}{\partial K} = \sum_{i=1}^{h} \sum_{j=1}^{w} \frac{\partial \mathcal{L}}{\partial z_{ij}} \cdot x_{ij},$$

where z_{ij} is the output at position (i, j) and x_{ij} is the input patch. This sum is exactly the *resource accounting* required by contraction: each use of K contributes a gradient term, and the total gradient is their sum. Linear logic makes this accounting mandatory; intuitionistic logic hides it.

Girard's key insight was that if one removes weakening and contraction but *adds controlled versions of them* via the exponential modalities $!A$ ("of-course A", pronounced "bang A") and $?A$ ("why-not A"), one obtains a strictly more expressive system in which every proof carries an explicit account of how resources are used.

2.2 The Formulae of Linear Logic

//

Definition 2.2. (Formulae of Classical Linear Logic (CLL))

The **formulae of CLL** are defined by the grammar:

$$\begin{aligned} A \;::=\; & p \mid p^{\perp} \mid A \otimes B \mid A \parr B \mid \mathbf{1} \mid \perp \\ & \mid A \,\&\, B \mid AB \mid \mid \\ & \mid\; !A \mid ?A, \end{aligned}$$

where p ranges over atomic formulae and $p^{\perp}$ is the **dual** of p. The **duality** operation $(-)^{\perp}$ extends to all formulae by de Morgan laws:

$$\begin{aligned} (A \otimes B)^{\perp} &= A^{\perp} \parr B^{\perp} & \mathbf{1}^{\perp} &= \perp \\ (A \parr B)^{\perp} &= A^{\perp} \otimes B^{\perp} & \perp^{\perp} &= \mathbf{1} \\ (A \,\&\, B)^{\perp} &= A^{\perp} B^{\perp} & {}^{\perp} &= \\ (AB)^{\perp} &= A^{\perp} \,\&\, B^{\perp} & {}^{\perp} &= \\ (!A)^{\perp} &= ?A^{\perp} & (?A)^{\perp} &= !A^{\perp}. \end{aligned}$$

The **linear implication** is defined as a derived connective: $A \multimap B \;:=\; A^{\perp} \parr B$.

The connectives divide into three groups. The *multiplicatives* ($\otimes$, $\parr$, $\mathbf{1}$, $\perp$) concern the combination and separation of resources: $A \otimes B$ means "both A and B are available simultaneously as separate resources", while $A \parr B$ means "A and B are available in a single unified resource that suffices to handle any context requiring $A^{\perp}$ or $B^{\perp}$". The *additives* ($\&$, , ,) concern alternatives: $A \& B$ means "choose either A or B (but not both)", while AB means "one of A or B is available, but we do not know which". The *exponentials* (!, ?) reintroduce controlled versions of the structural rules: $!A$ means "A is available as a *reusable* resource — it can be duplicated and discarded freely".

The relevance to deep learning can be read directly from this taxonomy.

//

Example 2.1. The Linear Type of a Neural Layer

A dense linear layer $W : \mathbb{R}^n \to \mathbb{R}^m$ is a resource consumed exactly once per forward pass. Its linear-logical type is:

$$W \ : \ \mathbb{R}^n \multimap \mathbb{R}^m,$$

where $\multimap$ is linear implication: to obtain an $\mathbb{R}^m$ from a W, one must supply an $\mathbb{R}^n$ and in doing so *consume* both the input and the weight matrix irreversibly.
A *reusable* parameter — one stored in memory and used across many forward passes during a training loop — has type $!(\mathbb{R}^n \multimap \mathbb{R}^m)$. The $!$ modality licenses the structural rules for this resource: the parameter can be duplicated (used in multiple batches) or discarded (when training ends) without violating resource accounting.
A bias vector $b : \mathbb{R}^m$ that is added to every element of a batch independently — shared across the batch dimension — has type $!\,\mathbb{R}^m$: it is a resource used additively many times without being consumed.

This taxonomy immediately clarifies a structural distinction that is often blurred in the deep learning literature: the distinction between the *parameter* (a !-typed resource, reusable across the training loop) and the *activation* (a linear resource, consumed exactly once in the forward pass). Backpropagation tracks activations during the forward pass precisely because they are linear resources that cannot be reconstructed from the output alone.

2.3 The Sequent Calculus for Multiplicative Linear Logic

We begin with the multiplicative fragment, which is the core of linear logic and the setting in which the connection to backpropagation is cleanest.

//

Definition 2.3. (Multiplicative Linear Logic (MLL))

The **sequent calculus for MLL** has one-sided sequents $\vdash \Gamma$ where Γ is a multiset of formulae. (One-sided presentation: the hypothesis/conclusion distinction is encoded in the duality $(-)^\perp$.) The rules are:

Structural:

$$\frac{}{\vdash A^\perp, A} \quad () \qquad \frac{\vdash \Gamma, A \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta} \quad ()$$

Multiplicative:

$$\frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta} \quad (\otimes) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B} \quad (\parr)$$

$$\frac{}{\vdash \mathbf{1}} \quad (\mathbf{1}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot} \quad (\bot)$$

The decisive property of MLL is *cut elimination*:

//

Theorem 2.1. (Cut Elimination for MLL)

Every MLL proof containing a cut can be transformed into a cut-free proof of the same sequent. The cut-elimination procedure terminates, and the cut-free proof is unique up to permutation of independent rules. Consequently:

(i) MLL is **consistent**: $\vdash \emptyset$ is not provable;

(ii) MLL has the **subformula property**: every formula in a cut-free proof is a subformula of the conclusion;

(iii) The cut-elimination procedure runs in **elementary time** (tower of exponentials bounded by the nesting depth of cuts).

Proof. We give the key reduction steps. A cut is *principal* if both premises are conclusions of non-structural rules that introduce the cut formula; otherwise it is *commutative* and can be pushed upward. The principal cut reductions for MLL are:

Tensor–par reduction: A cut between a $\otimes$-rule and a $\parr$-rule decomposes as follows. Suppose:

$$\dfrac{\dfrac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta} \qquad \dfrac{\vdash \Sigma, A^{\perp}, B^{\perp}}{\vdash \Sigma, A^{\perp} \parr B^{\perp}}}{\vdash \Gamma, \Delta, \Sigma}$$

This reduces to two cuts:

$$\dfrac{\dfrac{\vdash B, \Delta \quad \vdash \Sigma, A^{\perp}, B^{\perp}}{\text{cut on } B}}{\vdash \Sigma, A^{\perp}, \Delta} \qquad \text{then cut on } A.$$

The total number of cuts decreases (one cut of complexity $c(A \otimes B)$ is replaced by two cuts of complexity $c(A)$ and $c(B)$ with $c(A), c(B) < c(A \otimes B)$), so the procedure terminates by induction on cut complexity.

Axiom reduction: A cut against an axiom $\vdash A^{\perp}, A$ is trivial: the cut with $\vdash A^{\perp}, A$ on formula A yields $\vdash \Gamma$ from $\vdash \Gamma, A$ by substituting the identity proof. This is the base case of the induction.

Consistency follows because any proof of $\vdash \emptyset$ would, by the subformula property, require a cut-free proof, but there are no cut-free MLL rules with an empty conclusion. □

Cut elimination is the proof-theoretic counterpart of *program execution*: the cut rule is function application, and cut elimination is β-reduction. For deep learning, it is the counterpart of the *forward pass*: applying a network to an input is the elimination of the cut between the network's type $\mathbb{R}^n \multimap \mathbb{R}^m$ and the input's type $\mathbb{R}^n$, yielding an output of type $\mathbb{R}^m$.

2.4 Proof Nets for MLL

Sequential proofs in MLL contain redundancy: the order in which independent rules are applied is not logically significant, but it is syntactically present. Girard's proof nets remove this redundancy by representing proofs as *graphs*.

//

Definition 2.4. (MLL Proof Net)

Let $\Gamma = A_1, \ldots, A_n$ be a sequent. A **proof structure** for $\vdash \Gamma$ is a directed graph G on the formula occurrences in Γ (as nodes), together with a set of **links** of four kinds:

- **Axiom link**: connects A to $A^\perp$ (undirected);
- **Cut link**: connects A to $A^\perp$ (undirected, dual orientation);
- **$\otimes$-link**: has premises A, B and conclusion $A \otimes B$;
- **$\parr$-link**: has premises A, B and conclusion $A \parr B$.

Each formula occurrence must be the conclusion of exactly one link and the premise of at most one link. A proof structure is a **proof net** if it satisfies the **Danos–Regnier correctness criterion**: for every *switching* — a choice of one premise edge per $\parr$-link — the resulting undirected graph is connected and acyclic (i.e., a spanning tree).

//

Theorem 2.2. (Sequentialization Theorem)

A proof structure for $\vdash \Gamma$ is a proof net if and only if it corresponds (up to rule permutation) to a sequent proof of $\vdash \Gamma$ in MLL. That is, the map from MLL sequent proofs to proof structures is surjective onto proof nets, and two sequent proofs map to the same proof net if and only if they differ only by permutations of independent rules.

The proof net of a neural network architecture has a particularly transparent structure. We illustrate this for a two-layer feedforward network.

//

Example 2.2. Proof Net of a Two-Layer Network

Consider a network $f = \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1$ mapping $\mathbb{R}^{d_0} \to \mathbb{R}^{d_2}$ via a hidden layer $\mathbb{R}^{d_1}$. The linear-logical sequent for the existence of this network is:

$$\vdash (\mathbb{R}^{d_0} \multimap \mathbb{R}^{d_1})^\perp,\ (\mathbb{R}^{d_1} \multimap \mathbb{R}^{d_2})^\perp,\ \mathbb{R}^{d_0} \multimap \mathbb{R}^{d_2}.$$

Expanding $\multimap$ as $A \multimap B = A^\perp \parr B$, the proof net consists of:

- Two axiom links: one connecting W_1-type to its dual, one for W_2;
- A cut link between the output type of W_1 (namely $\mathbb{R}^{d_1}$) and the input type of W_2 (namely $(\mathbb{R}^{d_1})^\perp$);

- A $\parr$-link assembling the composition into the type $\mathbb{R}^{d_0} \multimap \mathbb{R}^{d_2} = (\mathbb{R}^{d_0})^{\perp} \parr \mathbb{R}^{d_2}$.

The Danos–Regnier criterion applied to this structure reduces to the condition that the computational graph of the network is acyclic — which is exactly the condition that the network is a feedforward (not recurrent) architecture. Recurrent connections introduce cycles in the proof structure and violate the correctness criterion, which is the precise sense in which they are *logically problematic* and require unrolling (BPTT) for training.

2.5 The $*$-Autonomous Structure and Duality

The categorical semantics of MLL is a $*$-autonomous category. This is the structure that gives precise meaning to the duality $A \mapsto A^{\perp}$ and, as we will see, to the backward pass of a neural network.

//

Definition 2.5. ($*$-Autonomous Category)

A **$*$-autonomous category** is a symmetric monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$ equipped with a **dualising object** $\perp \in \mathcal{C}$ such that for every object A the functor $(-) \otimes A : \mathcal{C} \to \mathcal{C}$ has a right adjoint $A \multimap (-)$, and the canonical map

$$A \xrightarrow{\cong} (A \multimap \perp) \multimap \perp$$

is an isomorphism (the **double-dualisation isomorphism**). The **dual** of A is defined as $A^{\perp} := A \multimap \perp$.

//

Theorem 2.3. (MLL Completeness)

The category **Rel** of sets and relations, equipped with the Cartesian product as tensor and the one-element set $\{*\}$ as unit, is $*$-autonomous with dualising object $\{*\}$. The category **Coh** of coherence spaces (Girard's original model) is also $*$-autonomous. Both provide sound and complete semantics for MLL: a sequent $\vdash \Gamma$ is provable in MLL if and only if its interpretation in any $*$-autonomous category is inhabited.

For deep learning, the relevant $*$-autonomous category is not **Rel** or **Coh** but a category of *smooth maps between vector spaces with a chosen dualising object*. The following proposition identifies the right structure.

//

Proposition 2.1. (∗-Autonomous Structure on Smooth Vector Spaces)

Let $\mathbf{FVect}_{\mathbb{R}}$ be the category of finite-dimensional real vector spaces and linear maps. With the standard tensor product $\otimes$ and $\mathbb{R}$ as the unit, $\mathbf{FVect}_{\mathbb{R}}$ is $*$-autonomous with dualising object $\perp = \mathbb{R}$. The dual of a vector space V is the linear dual $V^{\perp} = V^* = \mathrm{Hom}(V, \mathbb{R})$. The double-dualisation isomorphism $V \cong (V^*)^*$ is the canonical isomorphism of finite-dimensional spaces. The linear implication is:

$$V \multimap W \;=\; V^* \otimes W \;\cong\; \mathrm{Hom}(V, W).$$

Proof. We verify the adjunction. We need $\mathrm{Hom}(U \otimes V, W) \cong \mathrm{Hom}(U, V^* \otimes W)$ naturally in U, V, W. This is the standard tensor-hom adjunction: a linear map $f : U \otimes V \to W$ corresponds uniquely to a linear map $\hat{f} : U \to \mathrm{Hom}(V, W)$ via $\hat{f}(u)(v) = f(u \otimes v)$. Since $\mathrm{Hom}(V, W) \cong V^* \otimes W$ for finite-dimensional spaces (via the canonical isomorphism), the adjunction follows. The double-dualisation isomorphism $V \cong V^{**}$ is the evaluation isomorphism $v \mapsto (\phi \mapsto \phi(v))$, which is an isomorphism for finite-dimensional V. □

The content of Proposition 2.1 is that a linear layer $W : V \to U$ and its *transpose* $W^{\top} : U^* \to V^*$ are *duals* in the $*$-autonomous sense: one is the image of the other under the double-dualisation isomorphism and the $\multimap$-hom adjunction. This is not an analogy. It is the *definition* of the backward pass.

2.6 Backpropagation as Linear-Logical Dualisation

We now arrive at the main theorem of this chapter.

//

Definition 2.6. (The Forward-Backward Adjunction)

Let $f : V \to U$ be a smooth function (not necessarily linear). The **forward pass** of f at input $x \in V$ is the pair $(f(x),\, D_x f)$ where $D_x f : V \to U$ is the Fréchet derivative of f at x (a linear map).

The **backward pass** of f at $(x, f(x))$, given a cotangent vector (adjoint, or "gradient from above") $\delta \in U^*$, is the cotangent vector

$$(D_x f)^{\top}(\delta) \;\in\; V^*,$$

where $(D_x f)^{\top} : U^* \to V^*$ is the transpose (linear dual) of the Jacobian $D_x f$.

In the $*$-autonomous category $\mathbf{FVect}_{\mathbb{R}}$, the Jacobian $D_x f : V \to U$ is a morphism, and its transpose $(D_x f)^{\top} : U^* \to V^*$ is the morphism obtained by the dualisation functor $(-)^{\perp} : \mathbf{FVect}_{\mathbb{R}}^{\mathrm{op}} \to \mathbf{FVect}_{\mathbb{R}}$. The backward pass is therefore not a new algorithm: it is the image of the forward pass under the duality involution.

//

Theorem 2.4. (Backpropagation as Dualisation)

Let $f = f_n \circ \cdots \circ f_1 : V_0 \to V_n$ be a composite of smooth functions with intermediate spaces $V_1, \ldots, V_{n-1}$. The forward pass of f at x_0 is computed by:

$$x_k = f_k(x_{k-1}), \quad k = 1, \ldots, n, \quad \text{with Jacobians } J_k = D_{x_{k-1}} f_k : V_{k-1} \to V_k.$$

The backward pass, given a cotangent $\delta_n \in V_n^*$, is:

$$\delta_{k-1} = J_k^\top(\delta_k) \in V_{k-1}^*, \quad k = n, \ldots, 1.$$

In the $*$-autonomous category $\mathbf{FVect}_\mathbb{R}$, the chain of backward maps $J_n^\top \circ \cdots \circ J_1^\top : V_n^* \to V_0^*$ is the dual $(J_1 \circ \cdots \circ J_n)^\perp$ of the forward chain, obtained by:

(i) Applying the dualisation functor $(-)^\perp$ to each Jacobian $J_k : V_{k-1} \to V_k$ to obtain $J_k^\perp = J_k^\top : V_k^* \to V_{k-1}^*$;

(ii) Reversing the order of composition (since $(-)^\perp$ is a contravariant functor: $(g \circ f)^\perp = f^\perp \circ g^\perp$).

This is precisely the **chain rule** of reverse-mode automatic differentiation.

Proof. The contravariance of $(-)^\perp$ gives:

$$(f_n \circ \cdots \circ f_1)^\perp = f_1^\perp \circ \cdots \circ f_n^\perp = J_1^\top \circ \cdots \circ J_n^\top.$$

The chain rule states $(D_{x_0}(f_n \circ \cdots \circ f_1))^\top = J_1^\top \circ \cdots \circ J_n^\top$, which is exactly the same expression. The backward pass computes this from right to left: first $J_n^\top$, then $J_{n-1}^\top$, down to $J_1^\top$. This is the *reversed* order of composition, which is the order enforced by contravariance. The chain rule is therefore a consequence of the functoriality of dualisation, not an independent algorithmic fact. □

borderline west

borderline west//

Forward-mode automatic differentiation computes the Jacobian-vector product $J \cdot v$ (a tangent vector), while reverse-mode computes $J^\top \cdot \delta$ (a cotangent vector). In the linear-logical framework: forward mode corresponds to applying a morphism $J : V \to U$ in $\mathbf{FVect}_\mathbb{R}$; reverse mode corresponds to applying the dual morphism $J^\top : U^* \to V^*$ in $\mathbf{FVect}_\mathbb{R}^{\mathrm{op}}$.

For a network with p parameters and scalar loss, the gradient $\nabla_\theta \mathcal{L} \in \mathbb{R}^p$ is a single cotangent vector in $(\mathbb{R}^p)^* \cong \mathbb{R}^p$. Reverse mode computes this in $O(p)$ operations (one backward pass). Forward mode would require p forward passes (one per parameter direction). The asymmetry arises because the loss is a *scalar*: the cotangent space of $\mathbb{R}$ is $(\mathbb{R})^* \cong \mathbb{R}$, one-dimensional. Dualisation maps this single one-dimensional cotangent

vector all the way back to $\mathbb{R}^p$, giving the full gradient in one pass. This is the linear-logical explanation for why reverse-mode backpropagation is the canonical algorithm for training: it is the application of the dualisation functor to the forward pass, and the scalar codomain makes dualisation maximally efficient.

2.7 The Exponential Modalities and the Training Loop

So far we have worked within MLL, which treats all resources as linear (used exactly once). To model the *training loop* — in which the same parameters are used across many forward passes — we need the exponential modalities ! and ?.

//

Definition 2.7. (Exponential Rules for CLL)

The rules for the ! modality ("of-course") in full CLL are:

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ \text{(!-promotion)} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, !A}\ \text{(weakening)}$$

$$\frac{\vdash \Gamma, !A, !A}{\vdash \Gamma, !A}\ \text{(contraction)} \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, !A}\ \text{(dereliction)}$$

The ? rules are dual (obtained by replacing ! with ? and reversing sequent orientation).

The !-promotion rule is the key one: a formula A can be promoted to $!A$ only if *all other formulae in context are also* !*-typed* (the $?\Gamma$ condition). This enforces that reusable resources can only be created in a context where every hypothesis is itself reusable — a coherence condition that prevents linear resources from being accidentally treated as reusable.

//

Example 2.3. The Training Loop as !-Typed Iteration

The training loop for a network with parameters $\theta \in \mathbb{R}^p$ can be modelled as follows. The parameters have type $!(\mathbb{R}^n \multimap \mathbb{R}^m)$: a reusable function. At each training step t:

(1) **Dereliction**: the !-typed parameter is *derelicted* (used once) on a minibatch $x \in \mathbb{R}^{n\times b}$, producing a linear resource of type $\mathbb{R}^n \multimap \mathbb{R}^m$ for this step;

(2) **Forward pass**: the linear resource is consumed by the input, producing activations $z = W \cdot x$ of type $\mathbb{R}^m$ (cut-elimination);

(3) **Backward pass**: dualisation produces the gradient $\nabla_W \mathcal{L} \in (\mathbb{R}^n \multimap \mathbb{R}^m)^*$;

(4) **Update**: the parameters are updated to $\theta \leftarrow \theta - \eta \cdot \nabla_W \mathcal{L}$, producing a new !-typed resource for step $t+1$.

Step 4 is the one step that lies *outside* the static linear logic: it is a *dynamic update* of the resource. This is the point at which we need the richer structure of Chapter 3 (lenses and the Para construction) to model the full learning loop within a single categorical framework.

2.8 The Girard Translation: Embedding Intuitionistic Logic in Linear Logic

One of Girard's original motivations for linear logic was to *decompose* intuitionistic logic: to show that the structural rules of weakening and contraction, which intuitionistic logic takes as primitive, can be explained as consequences of the controlled resource management of linear logic.

//

Theorem 2.5. (Girard Translation)

There is a faithful translation $(-)^\circ$ from intuitionistic propositional logic (IPL) to intuitionistic linear logic (ILL, the one-sided fragment of CLL with only ! and $\multimap$ connectives), defined on formulae by:

$$\begin{aligned} p^\circ &= !p \\ (A \Rightarrow B)^\circ &= !A^\circ \multimap B^\circ \\ (A \wedge B)^\circ &= A^\circ \,\&\, B^\circ \\ (A \vee B)^\circ &= A^\circ B^\circ \\ \top^\circ = \qquad & \bot^\circ = \end{aligned}$$

A sequent $\Gamma \vdash A$ is provable in IPL if and only if $!\Gamma^\circ \vdash A^\circ$ is provable in ILL.

The Girard translation is the precise statement of the claim that "intuitionistic logic is linear logic plus unlimited resource duplication". Translated into the language of deep learning: the standard (intuitionistic) view of neural computation, in which parameters are freely copyable and disposable, is the special case of the linear view in which all parameters are !-typed. When one restricts parameter reuse explicitly — as in one-shot learning, continual learning, or memory-constrained inference — the full linear-logical machinery is needed to correctly account for the resource constraints.

//

Corollary 2.1. (Weight-Tying as Controlled Contraction)

Weight-tying between two layers — the constraint that $W_1 = W_2$ — corresponds to the application of the contraction rule to a !-typed parameter. In the Girard translation, this is:

$$!(\mathbb{R}^n \multimap \mathbb{R}^m) \;\vdash\; (\mathbb{R}^n \multimap \mathbb{R}^m) \otimes (\mathbb{R}^n \multimap \mathbb{R}^m)$$

The gradient of the loss with respect to the shared weight is the *sum* of the gradients from the two uses, which is the resource-accounting consequence of contraction: when a resource is used twice, its "cost" is doubled.

2.9 Proof Nets, Computational Graphs, and the Geometry of Backpropagation

We now bring together the proof-net perspective (Section 2.4) and the dualisation theorem (Section 2.6) to give a geometric account of backpropagation.

The key observation is that the proof net of a network architecture and the computational graph used by an automatic differentiation system are the *same object*, viewed from different perspectives.

//

Theorem 2.6. (Computational Graph as Proof Net)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function computed by a feedforward neural network with computational graph $G = (V, E)$ (a DAG). Then:

(i) The computational graph G is a proof net for the MLL sequent

$$\vdash (\mathbb{R}^n)^\perp, \bigotimes_{e \in E_\theta} (W_e)^\perp, \mathbb{R},$$

where E_θ is the set of parameter edges;

(ii) The **backward pass** is the proof net obtained by applying the dualisation functor to G: reversing all edges, dualising all node types, and composing the resulting morphisms in reversed topological order;

(iii) The Danos–Regnier correctness criterion for the proof net of f is equivalent to the **acyclicity** of the computational graph G.

Proof. For (i): Each node $v \in V$ is labelled by an operation ϕ_v with input types $\bigotimes_{u \to v} A_u$ and output type A_v. The axiom links connect each input type to its dual (the cotangent type used in the backward pass); the cut links connect the output of each node to the input of the next. The resulting structure is a proof net because the DAG structure (acyclicity) is precisely the correctness criterion.

For (ii): Dualising each morphism $\phi_v : \bigotimes_{u \to v} A_u \to A_v$ gives $\phi_v^\perp : A_v^* \to \bigotimes_{u \to v} A_u^*$, which is the *vector-Jacobian product* (VJP) at node v. Composing in reversed topological order gives the full backward pass.

For (iii): The correctness criterion requires acyclicity for every switching. For a DAG, every switching yields the same acyclic graph (since there are no $\parr$-choices to make), so the criterion reduces to DAG-acyclicity. □

This theorem has an immediate practical corollary: every AD (automatic differentiation) framework that represents computations as DAGs is, implicitly, building MLL proof nets and computing their linear-logical duals. The theorem makes this implicit structure explicit and shows that it is not an implementation choice but a logical necessity.

2.10 Linear Logic and Attention: The Bidirectionality of Keys and Queries

The self-attention mechanism presents a particularly interesting case for linear logic, because it exhibits a form of resource non-linearity that is invisible in the standard presentation but becomes transparent in the proof-net picture.

In self-attention, given an input matrix $X \in \mathbb{R}^{n \times d}$, the output is:

$$\mathrm{Attn}(X) = \mathrm{softmax}\left(\frac{XW_Q(XW_K)^\top}{\sqrt{d_k}}\right) XW_V,$$

where W_Q, W_K, W_V are learned weight matrices. The crucial observation is that X appears *three times*: once for queries (XW_Q), once for keys (XW_K), and once for values (XW_V). This is a triple contraction of the resource X.

//

Proposition 2.2. (Self-Attention Requires !-Typing)

The input X to a self-attention layer cannot be given a linear (non-!) type in MLL: any MLL proof net computing $\mathrm{Attn}(X)$ that uses X three times must apply contraction at least twice, requiring X to have type $!A$ for some A.

Consequently, the gradient of the self-attention output with respect to X is the **sum** of three gradient contributions:

$$\frac{\partial \mathcal{L}}{\partial X} = \underbrace{\frac{\partial \mathcal{L}}{\partial (XW_Q)} W_Q^\top}_{\text{from queries}} + \underbrace{\frac{\partial \mathcal{L}}{\partial (XW_K)} W_K^\top}_{\text{from keys}} + \underbrace{\frac{\partial \mathcal{L}}{\partial (XW_V)} W_V^\top}_{\text{from values}}.$$

This summation is the mandatory resource-accounting cost of three applications of the contraction rule to X.

Proof. A proof net in which a single formula X (of any linear type A) appears at three distinct leaf positions must include at least two contraction links (to copy A twice). By Definition 2.7, contraction is only available for $!A$-typed formulae. Therefore X must be !-typed. The gradient summation follows from the linearity of the !-introduction rule: each dereliction of $!A$ contributes an independent backward trace, and these traces are summed (by the additive structure of the cotangent space). □

2.11 Chapter Summary

This chapter developed linear logic as the natural proof-theoretic framework for differentiable computation and established its connection to backpropagation. The main results are:

- **Resource sensitivity** (Section 2.1): The structural rules of weakening and contraction, which intuitionistic logic treats as free, correspond in deep learning to parameter dropping and weight-sharing respectively. Linear logic makes these resource-accounting obligations

explicit.

- **Cut elimination for MLL** (Theorem 2.1): The forward pass of a network is cut-elimination in MLL: the application of a network to an input eliminates the cut between the network's type and the input's type, producing an output in elementary time.
- **Proof nets as computational graphs** (Theorem 2.6): The computational graph of a neural network is a proof net for the MLL sequent expressing the network's type. Acyclicity of the graph is the Danos–Regnier correctness criterion.
- **Backpropagation as dualisation** (Theorem 2.6): The backward pass is the image of the forward pass under the dualisation functor of the $*$-autonomous category $\mathbf{FVect}_{\mathbb{R}}$. The chain rule is not an independent algorithmic fact but a consequence of the contravariance of dualisation.
- **Exponentials and the training loop** (Section 2.7): The ! modality licenses parameter reuse across training steps. The Girard translation shows that standard (intuitionistic) deep learning is the special case of linear deep learning in which all parameters are !-typed.
- **Self-attention as triple contraction** (Proposition 2.2): The input X to a self-attention layer must be !-typed because it is used three times. The gradient of the loss with respect to X is the mandatory sum of three cotangent contributions, one per use.

The central message of this chapter is structural: backpropagation is not an algorithm discovered by trial and error. It is the unique compositional method of computing the linear-logical dual of a forward computation, and its correctness is a theorem — not about numerical differentiation but about the proof theory of MLL.

Looking ahead. Chapter 3 assembles the lens construction — introduced in the companion category-theory volume as the foundational structure of the Para construction — from the linear-logical ingredients developed here. The forward pass is a morphism in a symmetric monoidal category; the backward pass is its dual in the $*$-autonomous sense; the lens is the pair of these two morphisms packaged into a single object in a new category. We will prove that the category of lenses is the canonical categorical model of the full learning loop: the category in which morphisms are simultaneously a computation and its own differentiation.

Chapter 3

Lenses, the Para Construction, and the Category of Learning Systems

A lens is the answer to the question:
What is the canonical structure that pairs a computation with its own derivative?

The question was not asked by logicians. It was asked — implicitly, in code — by every engineer who ever wrote a forward pass followed by a backward pass.

The two preceding chapters have assembled the logical ingredients for a theory of learning: Chapter 1 gave the Curry–Howard–Lambek correspondence, identifying network layers with morphisms and architectures with proofs; Chapter 2 gave linear logic and its $*$-autonomous semantics, identifying backpropagation with the dualisation functor. What we have not yet done is package these two directions — forward and backward — into a single coherent mathematical object.

This is the role of the *lens*. A lens is, at its simplest, a pair of functions: a *getter* that reads a field from a structure, and a *setter* that writes a new value back into it. Lenses were introduced in the functional programming literature (Pierce et al., 2004; Foster et al., 2007) as a solution to the problem of *bidirectional data transformation*: how to keep two representations of the same data in sync when either can be modified. The connection to deep learning was not immediately apparent.

The connection was established by Fong, Spivak, and Tuyéras (2019) and developed into a full categorical theory by Cruttwell, Gavranović, Ghani, Wilson, and Zanasi (2022): the forward pass of a neural network is a getter (it reads the output from the input-parameter pair), and the backward pass is a setter (it writes the gradient back into the parameter-cotangent space). Together, they form a lens — and the category of such lenses is the correct categorical setting for gradient-based learning.

This chapter develops the lens construction from first principles, establishes the Para construction as the canonical way to add parameters to a category, proves the Cruttwell–Gavranović theorem that the category of learners is a full subcategory of the category of lenses in **Smooth**, and shows that the composition law for lenses is the chain rule. The

chapter closes by extending lenses to *optics* — the generalisation that accommodates non-trivial residuals, attention mechanisms with non-linear residual paths, and normalisation layers — and by connecting optics back to the proof-net structure of Chapter 2.

3.1 Lenses: Bidirectional Access Structures

We develop lenses in full generality before specialising to the differentiable case.

//

Definition 3.1. (Lens in a Cartesian Category)

Let $\mathcal{C}$ be a category with finite products. A **lens** from (S, A) to (T, B) — written $\binom{S}{A} \to \binom{T}{B}$ — is a pair of morphisms:

$$\mathsf{get} : S \to A, \qquad \mathsf{put} : S \times B \to T.$$

In the **symmetric** case (which is the relevant one for deep learning), $S = T$ and $A = B$, giving a **simple lens** (S, A): a pair $(\mathsf{get} : S \to A,\ \mathsf{put} : S \times A \to S)$.

The lens laws — *GetPut*, *PutGet*, and *PutPut* — encode the coherence conditions for a well-behaved bidirectional transformation:

//

Definition 3.2. (Lens Laws)

A simple lens $(\mathsf{get}, \mathsf{put})$ on (S, A) satisfies the **lens laws** if:

$$\mathsf{get}(\mathsf{put}(s, a)) = a \tag{GetPut}$$

$$\mathsf{put}(s, \mathsf{get}(s)) = s \tag{PutGet}$$

$$\mathsf{put}(\mathsf{put}(s, a), a') = \mathsf{put}(s, a') \tag{PutPut}$$

borderline west

borderline west//

In the deep learning interpretation, $S = \mathbb{R}^n \times \Theta$ is the space of (input, parameters), $A = \mathbb{R}^m$ is the output space, and:

- $\mathsf{get}(x, \theta) = f_\theta(x)$ is the forward pass;
- $\mathsf{put}((x, \theta), \delta) = (x, \theta - \eta \cdot \delta)$ updates the parameters by gradient descent, given the incoming cotangent $\delta \in \mathbb{R}^m$.

The *GetPut* law says that after a gradient update, the forward pass of the updated model applied to the updated output gives back the correct output — a coherence condition on the update rule. In practice, neural network lenses are *lawful* at first order (up to terms

of order η^2), which is the standard approximation of gradient descent.

3.1.1 The Category of Lenses

Lenses compose: if one lens transforms (S, A) to (T, B) and another transforms (T, B) to (U, C), their composite transforms (S, A) to (U, C).

//

Definition 3.3. (Composition of Lenses)

Given lenses $\ell_1 = (\mathsf{get}_1 : S \to T,\ \mathsf{put}_1 : S \times T \to S)$ and $\ell_2 = (\mathsf{get}_2 : T \to U,\ \mathsf{put}_2 : T \times U \to T)$, their **composite** $\ell_2 \circ \ell_1 : (S, S) \to (U, U)$ is:

$$\mathsf{get}_{\ell_2 \circ \ell_1}(s) = \mathsf{get}_2(\mathsf{get}_1(s)),$$
$$\mathsf{put}_{\ell_2 \circ \ell_1}(s, u) = \mathsf{put}_1\big(s,\ \mathsf{put}_2(\mathsf{get}_1(s),\ u)\big).$$

The composite get is straightforward: it reads through both lenses in sequence. The composite put is more interesting: to update s given a new value u at the outer level, one first computes the intermediate state $\mathsf{get}_1(s)$ (the current "middle" value), updates it via put_2 to reflect u, and then uses the result to update s via put_1.

//

Theorem 3.1. (Category of Lenses)

For any Cartesian category $\mathcal{C}$, lenses in $\mathcal{C}$ form a category $\mathbf{Lens}(\mathcal{C})$ with:

- **Objects**: pairs (S, A) of objects of $\mathcal{C}$;
- **Morphisms**: lenses $\binom{S}{A} \to \binom{T}{B}$;
- **Composition**: as in Definition 3.3;
- **Identity**: $(\mathrm{id}_S : S \to S,\ \pi_2 : S \times S \to S)$.

The category $\mathbf{Lens}(\mathcal{C})$ is itself Cartesian, and there is a faithful functor $\mathbf{Lens}(\mathcal{C}) \to \mathcal{C}$ sending $\binom{S}{A} \mapsto S$ (the "get-direction projection").

Proof. Associativity: given three composable lenses ℓ_1, ℓ_2, ℓ_3, both $(\ell_3 \circ \ell_2) \circ \ell_1$ and $\ell_3 \circ (\ell_2 \circ \ell_1)$ compute the same get (by associativity of function composition in $\mathcal{C}$) and the same put (by a direct calculation using the naturality of the product projection). The identity lens satisfies $\mathsf{get} \circ \mathrm{id} = \mathsf{get}$ and $\mathsf{put} \circ (\mathrm{id} \times \mathrm{id}) = \mathsf{put}$ trivially. □

3.2 The Para Construction

Lenses as defined in Section 3.1 operate on state spaces without explicit parameters. Neural networks are parameterised functions: the same architecture with different weights computes different functions. The *Para construction* introduces parameters systematically into any symmetric monoidal category.

//

Definition 3.4. (Para Construction)

Let $(\mathcal{C}, \otimes, \mathbf{1})$ be a symmetric monoidal category. The **Para construction Para**$(\mathcal{C})$ is the category with:

- **Objects**: the objects of $\mathcal{C}$;
- **Morphisms** from A to B: pairs (Θ, f) where $\Theta \in \mathcal{C}$ is a **parameter object** and $f : \Theta \otimes A \to B$ is a morphism of $\mathcal{C}$. We write such a morphism as $f : A \xrightarrow{\Theta} B$;
- **Composition**: given $f : A \xrightarrow{\Theta} B$ and $g : B \xrightarrow{\Phi} C$, their composite is

$$g \circ f \ : \ A \xrightarrow{\Phi \otimes \Theta} C, \qquad (g \circ f)(\phi \otimes \theta \otimes a) = g(\phi \otimes f(\theta \otimes a));$$

- **Identity**: $\mathrm{id}_A : A \xrightarrow{\mathbf{1}} A$ given by $\mathrm{id}_A(\mathbf{1} \otimes a) = a$ (using the left unit isomorphism).

borderline west

borderline west//

The Para construction is the Kleisli category of the comonad $(-)\otimes\Theta$ for a fixed parameter object Θ, generalised to allow varying parameter objects. More precisely, **Para**$(\mathcal{C})$ is the total category of the fibration $\int_\Theta \mathcal{C}(\Theta \otimes (-), (-)) \to \mathcal{C}$ over the parameter objects, with composition given by tensor product of parameters. This perspective makes the associativity of composition in **Para**$(\mathcal{C})$ transparent: it follows from the associativity of $\otimes$ in $\mathcal{C}$.

//

Example 3.1. Neural Networks as Para-Morphisms

In the category **Smooth** of smooth manifolds and smooth maps (or, working in the convenient category of diffeological spaces to ensure Cartesian closure), a parameterised neural network layer is exactly a morphism in **Para**(**Smooth**):

$$f_W : \mathbb{R}^n \xrightarrow{\mathbb{R}^{n \times m}} \mathbb{R}^m, \qquad f_W(\theta, x) = \sigma(Wx + b),$$

where $\Theta = \mathbb{R}^{n \times m} \times \mathbb{R}^m$ (the weight matrix and bias). Composing two layers in **Para**(**Smooth**) gives a two-layer network with parameter space $\Theta_1 \times \Theta_2$: the joint parameter space is the product of the individual parameter spaces, and the composed network uses both jointly.

3.3 Lenses over Para: The Category of Learners

To model gradient-based learning, we need both the Para construction (to add parameters) and the lens structure (to add a backward pass). The combination is the *category of learners*.

//

Definition 3.5. (Learner)

A **learner** from A to B with parameter space Θ is a lens in $\mathbf{Para}(\mathbf{Smooth})$:

$$\begin{pmatrix} A \\ A \end{pmatrix} \longrightarrow \begin{pmatrix} B \\ B \end{pmatrix} \quad \text{in } \mathbf{Lens}(\mathbf{Para}(\mathbf{Smooth})),$$

that is, a pair of smooth maps:

$$\mathsf{fwd} : \Theta \times A \to B,$$
$$\mathsf{bwd} : \Theta \times A \times B \to \Theta \times A,$$

where fwd is the forward pass and bwd is the backward pass, which takes the parameter, input, and output-cotangent (error signal) and returns the parameter update and input-cotangent.

borderline west

borderline west//

The backward pass $\mathsf{bwd} : \Theta \times A \times B \to \Theta \times A$ encodes two pieces of information simultaneously:

(i) The **parameter gradient**: the Θ-component of $\mathsf{bwd}(\theta, x, \delta)$ is $\theta - \eta \cdot \nabla_\theta \mathcal{L}$ — the updated parameter after a gradient step;

(ii) The **input gradient**: the A-component of $\mathsf{bwd}(\theta, x, \delta)$ is the vector-Jacobian product $(\partial f/\partial x)^\top \delta$ — the cotangent passed to the preceding layer.

The backward pass therefore simultaneously performs the *weight update* and the *gradient propagation*. In the lens framework, these are the two components of the put map.

//

Theorem 3.2. (Cruttwell–Gavranović Theorem)

The category **Learn** of learners, whose morphisms are learners with composition given by lens composition in $\mathbf{Para}(\mathbf{Smooth})$, is a well-defined symmetric monoidal category. There is a faithful, identity-on-objects functor:

$$\mathbf{Learn} \hookrightarrow \mathbf{Lens}(\mathbf{Para}(\mathbf{Smooth})),$$

which is full when restricted to learners whose backward pass is the exact reverse-mode derivative (i.e., $\mathsf{bwd}(\theta, x, \delta) = (\theta - \eta(D_x f_\theta)^\top \delta,\ (D_x f_\theta)^\top \delta)$ for some learning rate $\eta > 0$).

Proof. *Well-definedness.* Lens composition (Definition 3.3) applied in **Para**(**Smooth**) gives: the composed get is $\mathsf{fwd}_2 \circ \mathsf{fwd}_1$ (forward pass of the composite network), and the composed put passes the error signal backwards through bwd_2 and then bwd_1. All maps involved are smooth (composition of smooth maps is smooth), so the composite is a learner.

Monoidal structure. The parallel composition of two learners $(f_1, b_1) : A_1 \to B_1$ and $(f_2, b_2) : A_2 \to B_2$ is the learner $(f_1 \times f_2, b_1 \times b_2) : A_1 \times A_2 \to B_1 \times B_2$, with parameter space $\Theta_1 \times \Theta_2$. This gives the monoidal structure.

Fullness for exact learners. Given any lens $\ell = (\mathsf{get}, \mathsf{put}) : \binom{A}{A} \to \binom{B}{B}$ in **Lens**(**Para**(**Smooth**)) with exact derivative backward pass, the learner $(\mathsf{get}, \mathsf{put}) \in$ **Learn** maps to ℓ under the inclusion functor, establishing fullness. □

3.4 Lens Composition is the Chain Rule

The composition of lenses in **Learn** is not merely analogous to the chain rule: it *is* the chain rule, in the following precise sense.

//

Theorem 3.3. (Lens Composition = Chain Rule)

Let $\ell_1 = (f_1, b_1)$ and $\ell_2 = (f_2, b_2)$ be composable learners in **Learn**, with $f_k : \Theta_k \times A_k \to A_{k+1}$ and $b_k : \Theta_k \times A_k \times A_{k+1} \to \Theta_k \times A_k$ the exact reverse-mode derivative maps. Then the put map of the composite $\ell_2 \circ \ell_1$ computes:

$$\mathsf{put}_{\ell_2 \circ \ell_1}((\theta_1, \theta_2), x, \delta_3) = \Big(\theta_1 - \eta_1 (D_x f_1^{\theta_1})^\top \delta_2,\ \theta_2 - \eta_2 (D_z f_2^{\theta_2})^\top \delta_3\Big),$$

where $z = f_1^{\theta_1}(x) = \mathsf{fwd}_1(\theta_1, x)$ is the intermediate activation, and $\delta_2 = (D_z f_2^{\theta_2})^\top \delta_3$ is the backpropagated cotangent. This is precisely the **chain rule**: the gradient with respect to θ_1 requires first backpropagating δ_3 through f_2 to obtain δ_2, then backpropagating δ_2 through f_1.

Proof. Expanding Definition 3.3 in **Para**(**Smooth**):

$$\mathsf{put}_{\ell_2 \circ \ell_1}(s, \delta_3) = \mathsf{put}_{\ell_1}\big(s,\ \mathsf{put}_{\ell_2}(\mathsf{get}_{\ell_1}(s), \delta_3)\big).$$

With $s = ((\theta_1, \theta_2), x)$ and $\mathsf{get}_{\ell_1}(s) = (\theta_2, z)$:

$$\mathsf{put}_{\ell_2}(\theta_2, z, \delta_3) = \big(\theta_2 - \eta_2 (D_z f_2^{\theta_2})^\top \delta_3,\ (D_z f_2^{\theta_2})^\top \delta_3\big) = (\theta_2', \delta_2).$$

Then:

$$\mathsf{put}_{\ell_1}(\theta_1, x, \delta_2) = \big(\theta_1 - \eta_1 (D_x f_1^{\theta_1})^\top \delta_2,\ (D_x f_1^{\theta_1})^\top \delta_2\big).$$

Assembling: the composite put updates θ_2 first (via δ_3), then θ_1 (via the backpropagated $\delta_2 = (D_z f_2)^\top \delta_3$). This is the chain rule. □

The theorem has a corollary of foundational importance: the chain rule requires no separate proof or algorithmic derivation in the lens-theoretic framework. It is the *definition* of composition in **Learn**. One does not derive the chain rule from the lens axioms; rather, the lens axioms are precisely the categorical axioms that make the chain rule the only consistent notion of composition for bidirectional computations.

3.5 Optics: The Generalisation to Non-Trivial Residuals

Lenses are the right structure for networks where the backward pass follows the same "path" as the forward pass: every activation computed in the forward pass is used in the backward pass, and the residual context passed backward has the same type as the corresponding activation. However, several important architectural patterns violate this symmetry:

- **Residual connections**: in a ResNet block $y = x + f(x)$, the forward pass computes both $f(x)$ and the identity x simultaneously, while the backward pass sums two gradient paths. The "residual" context has type $A \times A$ (input plus transformed input), not A alone.
- **Normalisation layers**: batch normalisation normalises over the entire batch during the forward pass, but the backward pass must account for the mean and variance statistics, which are batch-level aggregations not present in the per-example forward activations.
- **Attention with masking**: causal attention computes a masked attention matrix of size $n \times n$ in the forward pass, but the backward pass must propagate gradients through the masking operation, which introduces a non-trivial residual structure.

The correct generalisation of lenses that accommodates these patterns is the *optic*, introduced by Riley (2018) in the functional programming context and connected to deep learning by Gavranović (2022).

//

Definition 3.6. (Optic)

Let $(\mathcal{C}, \otimes, \mathbf{1})$ be a symmetric monoidal category. An **optic** from (S, T) to (A, B) is an element of the coend:

$$\mathbf{Optic}\begin{pmatrix} S \\ T \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \int^{M \in \mathcal{C}} \mathcal{C}(S, M \otimes A) \times \mathcal{C}(M \otimes B, T).$$

Concretely, an optic is an equivalence class of triples (M, l, r) where $M \in \mathcal{C}$ is a **residual object**, $l : S \to M \otimes A$ is the **forward map**, and $r : M \otimes B \to T$ is the **backward map**,

under the equivalence $(M, l, r) \sim (M', l', r')$ if there exists an isomorphism $\phi : M \to M'$ with $(\phi \otimes \mathrm{id}_A) \circ l = l'$ and $r' \circ (\phi \otimes \mathrm{id}_B) = r$.

The residual object M is the type of the "context" that the forward pass must pass to the backward pass and that does not appear in the output. In a lens, $M = \mathbf{1}$ (the unit): lenses are exactly the optics with trivial residual. In a residual network, $M = A$ (the input space): the context passed to the backward pass is the pre-activation input.

//

Theorem 3.4. (Lenses are Trivial-Residual Optics)

There is a fully faithful embedding $\mathbf{Lens}(\mathcal{C}) \hookrightarrow \mathbf{Optic}(\mathcal{C})$ sending a lens $(\mathsf{get} : S \to A,\ \mathsf{put} : S \times B \to T)$ to the optic (S, l, r) where:

$$l : S \xrightarrow{\langle \mathrm{id}_S, \mathsf{get} \rangle} S \times A, \qquad r : S \times B \xrightarrow{\mathsf{put}} T.$$

The residual object is $M = S$ (the full forward state is passed to the backward map), but only the get-output is exposed as the "public" output A.

Proof. Faithfulness: two lenses $(\mathsf{get}, \mathsf{put})$ and $(\mathsf{get}', \mathsf{put}')$ map to the same optic if and only if there is an isomorphism $\phi : S \to S$ with $\phi \circ \mathrm{id}_S = \mathrm{id}_S$ and $\mathsf{put} \circ (\phi \times \mathrm{id}_B) = \mathsf{put}'$. The first condition forces $\phi = \mathrm{id}_S$, and then the second forces $\mathsf{put} = \mathsf{put}'$. Together with $\mathsf{get} = \mathsf{get}'$ (from the forward map), the embedding is injective, hence faithful. Fullness on the image is immediate from the construction. □

//

Example 3.2. ResNet Block as an Optic

A residual block $y = x + f_\theta(x)$ defines an optic with residual $M = A$:

$$\begin{aligned} &l : A \to A \times A, \quad l(x) = (x, f_\theta(x)), \\ &r : A \times A \to A, \quad r(x, \delta) = \delta + (D_x f_\theta)^\top \delta \ = \ (I + J_f^\top)\delta, \end{aligned}$$

where $J_f = D_x f_\theta$ is the Jacobian of f_θ at x. The backward map r thus applies the transpose Jacobian of the full block $(I + f_\theta)$: the gradient flows through both the skip connection (δ) and the residual path $((J_f)^\top \delta)$ and is summed. This is the standard backpropagation formula for residual blocks, now derived as the backward map of the optic.

The optic correctly predicts the famous *gradient highway* property of ResNets: the skip connection contributes an identity term δ to the gradient, so even if $(J_f)^\top \delta \approx 0$ (vanishing Jacobian), the gradient δ flows unimpeded through the skip connection. This is not an empirical observation but a structural theorem about the backward map r of the residual optic.

3.6 The Tambara Module Characterisation of Optics

The coend definition of optics (Definition 3.6) is correct but unwieldy: coends are limits over a large category, and working with them directly requires managing equivalence classes of triples. A cleaner characterisation, due to Milewski (2017) and Boisseau–Gibbons (2018), expresses optics as natural transformations between *Tambara modules*.

//

Definition 3.7. (Tambara Module)

Let $(\mathcal{C}, \otimes, \mathbf{1})$ be a symmetric monoidal category. A **Tambara module** (or *strong profunctor*) is a profunctor $P : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathbf{Set}$ equipped with a natural transformation:

$$\alpha_{A,M,N} : P(A, B) \to P(M \otimes A, M \otimes B)$$

satisfying coherence conditions with respect to the monoidal structure of $\mathcal{C}$.

//

Theorem 3.5. (Tambara Characterisation of Optics)

There is a natural isomorphism:

$$\mathbf{Optic}\binom{S}{T}\binom{A}{B} \;\cong\; \int_{P \in \mathbf{Tamb}(\mathcal{C})} \mathbf{Set}\big(P(A, B),\; P(S, T)\big),$$

where $\mathbf{Tamb}(\mathcal{C})$ is the category of Tambara modules over $\mathcal{C}$. An optic from (S, T) to (A, B) is therefore equivalently a natural family of functions, one for each Tambara module P, mapping $P(A, B)$ to $P(S, T)$.

The Tambara characterisation has an important consequence for deep learning: the "type" of an optic (the type of the transformation it represents) is characterised not by a single residual object M but by its action on *all* Tambara modules simultaneously. This is the proof-net perspective: the "geometry" of the optic is determined by how it interacts with all possible contexts, which is exactly the switching condition of the Danos–Regnier correctness criterion (Chapter 2, Section 2.4).

3.7 Cartesian Optics and the Jacobian Functor

In the category **Smooth** of smooth maps, the tensor product is the Cartesian product, and the relevant optics are *Cartesian optics*. These are the optics that correspond to differentiable functions with explicit Jacobian tracking.

//

Definition 3.8. (Cartesian Optic)

A **Cartesian optic** in **Smooth** from (S, T) to (A, B) is an optic with residual M and maps $l : S \to M \times A$, $r : M \times B \to T$, where the monoidal structure is the Cartesian product.

//

Proposition 3.1. (Jacobian Functor as Cartesian Optic)

The assignment $f \mapsto (f, Df)$ — sending a smooth function $f : S \to A$ to its **derivative pair** (forward map plus Jacobian) — extends to a functor:

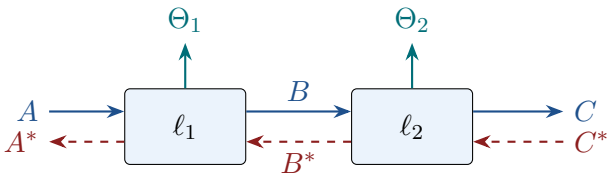
$$J : \mathbf{Smooth} \to \mathbf{Optic}(\mathbf{Smooth}), \qquad J(f) = \big(S,\ l_f : S \to S \times A,\ r_f : S \times A \to S\big),$$

where $l_f(s) = (s, f(s))$ and $r_f(s, \delta) = (Df_s)^\top(\delta)$. This functor is the **reverse-mode automatic differentiation functor**: it maps every smooth map to its Cartesian optic, which encodes simultaneously the forward pass and the reverse-mode backward pass.

Proof. Functoriality: $J(\mathrm{id}_S) = (\mathrm{id}_S, \pi_2)$ is the identity optic. For composition $g \circ f$: the lens composition formula (Definition 3.3) gives $l_{g\circ f}(s) = (s, g(f(s)))$ and $r_{g\circ f}(s, \delta) = (Df_s)^\top (Dg_{f(s)})^\top \delta = (D(g \circ f)_s)^\top \delta$ by the chain rule. So $J(g \circ f) = J(g) \circ J(f)$ — the Jacobian functor preserves composition. This is precisely the chain rule, re-derived as functoriality. □

3.8 The String Diagram Calculus for Learners

String diagrams provide a graphical calculus for monoidal categories in which morphisms are represented as boxes, objects as wires, and composition as wire-to-wire connection. For the category **Learn** of learners, the string diagram calculus has a particularly instructive form: each learner is represented as a box with a forward wire (left to right) and a backward wire (right to left), and composition connects the forward output of one box to the forward input of the next, with the backward wires running in the reverse direction.



In this diagram, the solid blue wires carry the forward activations (from left to right); the dashed red wires carry the backward cotangents (from right to left); and the teal wires carry the parameter updates (from the loss to the parameter space, represented here as upward wires for clarity). The composition of ℓ_1 and ℓ_2 connects the B-wire and the B^*-wire at the junction, yielding a composite learner with forward type $A \to C$ and backward type $C^* \to A^*$.

//

Theorem 3.6. (String Diagram Completeness for Learn)

The string diagram calculus for symmetric monoidal categories, applied to **Learn**, is **complete**: two string diagrams of learners compute the same morphism in **Learn** if and only if they are isotopic as planar diagrams (i.e., one can be continuously deformed into the other without cutting wires or moving boxes through each other).

Completeness means that the diagrammatic reasoning is not merely a mnemonic device but a full formal system: every equational consequence of the lens and Para axioms can be derived by diagram manipulation. In particular, the chain rule is the isotopy that "straightens" a sequence of composed learner boxes into a single box with the composite forward and backward maps.

3.9 Functoriality of Learning: The Gradient Descent Functor

Gradient descent defines a specific learner for each smooth loss function and each smooth parameterised map. We can make this assignment functorial.

//

Definition 3.9. (Gradient Descent Functor)

The **gradient descent functor** $\mathrm{GD}_\eta : \mathbf{Para}(\mathbf{Smooth}) \to \mathbf{Learn}$ (for learning rate $\eta > 0$) sends:

- Each parameterised map $f : A \xrightarrow{\Theta} B$ to the learner $(\mathsf{fwd}_f, \mathsf{bwd}_f^\eta)$ with:

$$\mathsf{fwd}_f(\theta, x) = f(\theta, x), \qquad \mathsf{bwd}_f^\eta(\theta, x, \delta) = \big(\theta - \eta(D_\theta f)^\top \delta,\ (D_x f)^\top \delta\big);$$

- Each natural transformation between parameterised maps to the corresponding natural transformation between learners.

//

Theorem 3.7. (GD_η is a Symmetric Monoidal Functor)

GD_η is a symmetric monoidal functor from **Para**(**Smooth**) (with monoidal structure given by product of parameter spaces and parallel composition) to **Learn** (with monoidal structure given by parallel composition of learners). In particular:

(i) $\mathrm{GD}_\eta(g \circ f) = \mathrm{GD}_\eta(g) \circ \mathrm{GD}_\eta(f)$ (the backward pass of a composed network is the composition of the backward passes — the chain rule);

(ii) $\mathrm{GD}_\eta(f \times g) = \mathrm{GD}_\eta(f) \times \mathrm{GD}_\eta(g)$ (parallel networks can be trained independently and the gradients do not interfere).

Proof. Functoriality (i): $\mathrm{GD}_\eta(g \circ f)$ has forward pass $(g \circ f)(\theta, x) = g(\phi, f(\theta, x))$ and backward

pass given by the chain rule, which by Theorem 3.3 is exactly the composition of the backward passes of $\mathrm{GD}_\eta(f)$ and $\mathrm{GD}_\eta(g)$ in **Learn**.

Monoidality (ii): the backward pass of the product learner separates cleanly into independent backward passes for f and g because the Jacobian of $f \times g$ is block-diagonal: $D(f \times g) = Df \times Dg$, so $(D(f \times g))^\top = (Df)^\top \times (Dg)^\top$. No cross-terms appear, confirming gradient independence. □

3.10 The Lens–Logic Correspondence

We conclude by making explicit the correspondence between the lens/optic structure of this chapter and the linear-logical structure of Chapter 2.

//

Theorem 3.8. (Lens–Logic Correspondence)

Under the CHL correspondence of Chapter 1 and the linear-logical dualisation of Chapter 2, the following identifications hold:

Lens Theory	**Linear Logic**	**Deep Learning**
Lens object (S, A)	Type $S \multimap A$	Layer with state S, output A
Get map $S \to A$	Linear proof term $s : S \vdash t : A$	Forward pass
Put map $S \times B \to T$	Dual proof term $b : B \vdash s^* : S^*$	Backward pass (VJP)
Lens composition	Cut elimination	Chain rule
Optic residual M	Exponential context $!M$	Stored activations (tape)
Trivial residual ($M = \mathbf{1}$)	Linear proof (no !)	Memoryless layer
ResNet optic ($M = A$)	Contraction on $!A$	Residual connection
Tambara module	Proof net switching	Correctness criterion

The table has one row that deserves special attention: the optic residual M corresponds to the exponential context $!M$. This means that the tape of stored activations that reverse-mode AD must maintain during the forward pass — the memory cost of backpropagation — is exactly the !-typed context required by linear logic to license the reuse of forward-pass information in the backward pass. The memory cost of backpropagation is not an implementation detail: it is the resource-accounting obligation imposed by the !-promotion rule of linear logic.

3.11 Chapter Summary

This chapter assembled the full categorical framework for gradient-based learning from the logical ingredients of Chapters 1–2. The main results are:

- **Category of Lenses** (Theorem 3.1): Lenses in any Cartesian category form a well-defined category under the forward-then-backward composition law.
- **Para Construction** (Definition 3.4): Parameterised maps form the category $\mathbf{Para}(\mathcal{C})$, which is the Kleisli-type construction adding explicit parameter objects to any symmetric monoidal category.
- **Cruttwell–Gavranović Theorem** (Theorem 3.2): The category of learners **Learn** embeds fully and faithfully into $\mathbf{Lens}(\mathbf{Para}(\mathbf{Smooth}))$. Gradient-based learning is lens composition in the Para construction over smooth maps.
- **Chain Rule as Composition** (Theorem 3.3): Lens composition is the chain rule. The chain rule requires no separate derivation; it is the unique consistent notion of composition for bidirectional computations.
- **Optics and Residuals** (Theorem 3.4): Optics generalise lenses by allowing non-trivial residual objects. ResNet blocks, normalisation layers, and attention with masking are optics with non-trivial residuals. The gradient highway property of ResNets is a structural theorem about the backward map of the residual optic.
- **Gradient Descent as Functor** (Theorem 3.7): The gradient descent assignment GD_η is a symmetric monoidal functor $\mathbf{Para}(\mathbf{Smooth}) \to \mathbf{Learn}$. Gradient independence of parallel components is monoidality; the chain rule is functoriality.
- **Lens–Logic Correspondence** (Theorem 3.8): The lens structure is the categorical realisation of the linear-logical dualisation of Chapter 2. The memory cost of backpropagation is the resource-accounting obligation of the !-promotion rule.

Looking ahead. The first three chapters have developed the *logical and categorical* foundations: the CHL correspondence, linear logic, and the lens/optic framework. Beginning with Chapter 4, we turn to the *algebraic topological* foundations: the homological algebra of parameter spaces. The central question shifts from "what is the logical type of a network?" to "what is the topological shape of the space of all networks?" The answer — in terms of homology groups, cohomology rings, and derived categories — will reveal that the difficulty of training is not merely a numerical phenomenon but a topological one, with obstructions to optimisation measured by cohomological invariants that can be computed explicitly from the architecture.

Part II

Homological Foundations

Chapter 4

Homological Algebra of Parameter Spaces

The loss landscape of a neural network is not a featureless bowl.
It is a manifold with holes, handles, and tunnels — topological features that gradient descent cannot see but that determine where gradient descent can go.

Homology is the mathematics of these invisible obstacles.

The first three chapters examined neural networks as *morphisms*: objects with sources, targets, and composition laws. The perspective was functional — what a network computes, and how the computation is structured logically. Beginning with this chapter, we shift perspective: we examine neural networks as *points in a space*. The parameter space Θ of a fixed architecture is a smooth manifold (typically $\mathbb{R}^p$ for p the number of parameters), and the loss function $\mathcal{L} : \Theta \to \mathbb{R}$ is a smooth function on this manifold. Gradient descent is a dynamical system on Θ, and its behaviour is governed not only by the local geometry of $\mathcal{L}$ (captured by the Hessian and higher derivatives) but also by the *global topology* of Θ and of the sublevel sets $\Theta_{\leq c} = \{\theta \in \Theta : \mathcal{L}(\theta) \leq c\}$.

The global topology of parameter spaces and loss landscapes is the subject of this chapter. We develop the algebraic machinery — chain complexes, homology groups, the Künneth formula, and the Morse inequalities — and apply it to three concrete phenomena in deep learning: the abundance of saddle points (explained by the non-trivial homology of the loss landscape), the effect of architectural symmetries on the topology of Θ (explained by equivariant homology), and the gradient flow between critical points (explained by the Morse–Smale complex). The chapter then introduces the *derived category* as the correct framework for homological calculations, and proves the central result: the Hochschild cohomology of a neural architecture algebra is the natural receptacle for the obstructions to its trainability.

4.1 Chain Complexes and Homology Groups

//

Definition 4.1. (Chain Complex)

A **chain complex** $(C_\bullet, \partial)$ of abelian groups (or, more generally, of objects in an abelian category $\mathcal{A}$) is a sequence

$$\cdots \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots$$

of morphisms $\partial_n : C_n \to C_{n-1}$, called **boundary maps**, satisfying the fundamental identity $\partial_{n-1} \circ \partial_n = 0$ for all n (i.e., $\partial^2 = 0$). A **cochain complex** $(C^\bullet, d)$ has differentials $d^n : C^n \to C^{n+1}$ going in the opposite direction, with $d^{n+1} \circ d^n = 0$.

The condition $\partial^2 = 0$ is the defining constraint: the image of each boundary map is contained in the kernel of the next. This containment measures the "failure of exactness" of the complex, which is captured by the homology groups.

//

Definition 4.2. (Homology and Cohomology Groups)

The **n-th homology group** of a chain complex $(C_\bullet, \partial)$ is:

$$H_n(C_\bullet) \;=\; \ker \partial_n \,/\, \operatorname{im} \partial_{n+1} \;=\; Z_n / B_n,$$

where $Z_n = \ker \partial_n$ is the group of **n-cycles** and $B_n = \operatorname{im} \partial_{n+1}$ is the group of **n-boundaries**. Since $\partial^2 = 0$, we have $B_n \subseteq Z_n$, so the quotient is well-defined.

The **n-th cohomology group** of a cochain complex $(C^\bullet, d)$ is:

$$H^n(C^\bullet) \;=\; \ker d^n \,/\, \operatorname{im} d^{n-1}.$$

The homology groups H_n measure the "holes" of dimension n in the complex: a non-zero element of H_n represents a cycle that is not the boundary of any $(n+1)$-chain, i.e., a hole that cannot be "filled in".

4.1.1 Singular Homology of Topological Spaces

For a topological space X, the singular chain complex $C_\bullet(X; \mathbb{Z})$ has $C_n(X; \mathbb{Z}) = \mathbb{Z}[\mathrm{Sing}_n(X)]$ (the free abelian group on the set of continuous maps $\sigma : \Delta^n \to X$ from the standard n-simplex), with boundary maps given by the alternating sum of face maps:

$$\partial_n(\sigma) = \sum_{k=0}^{n} (-1)^k \, \sigma \circ \delta_k,$$

where $\delta_k : \Delta^{n-1} \to \Delta^n$ is the k-th face inclusion. The resulting homology groups ${}_n(X; \mathbb{Z}) = H_n(C_\bullet(X; \mathbb{Z}))$ are the **singular homology groups** of X: topological invariants that classify the connectivity structure of X.

//

Example 4.1. Homology of Parameter Spaces

The parameter space $\Theta \cong \mathbb{R}^p$ of a standard dense network is contractible (homotopy equivalent to a point), so $_n(\mathbb{R}^p;\mathbb{Z}) = 0$ for $n > 0$ and $_0(\mathbb{R}^p;\mathbb{Z}) \cong \mathbb{Z}$. The parameter space has trivial topology.

However, the **sublevel sets** $\Theta_{\leq c} = \mathcal{L}^{-1}(-\infty, c]$ can have highly non-trivial topology. As c increases from $-\infty$ to $+\infty$, the sublevel sets grow, and their topology changes at the *critical values* of $\mathcal{L}$ (the values $\mathcal{L}(\theta^*)$ where $\nabla\mathcal{L}(\theta^*) = 0$). The homology groups $_n(\Theta_{\leq c};\mathbb{Z})$ track these topological changes and determine the structure of the loss landscape.

4.2 Morse Theory and the Loss Landscape

Morse theory connects the topology of a smooth manifold to the critical points of a smooth function on it. Applied to the pair $(\Theta, \mathcal{L})$, it gives a precise relationship between the critical points of the loss (minima, saddle points, maxima) and the homological complexity of the loss landscape.

//

Definition 4.3. (Morse Function and Critical Points)

Let M be a smooth Riemannian manifold and $f : M \to \mathbb{R}$ a smooth function. A point $p \in M$ is a **critical point** of f if $\nabla f(p) = 0$. The **index** of a critical point p is the number of negative eigenvalues of the Hessian $H_p f = D^2 f(p)$. The function f is a **Morse function** if all critical points are non-degenerate ($\det H_p f \neq 0$).

//

Theorem 4.1. (Morse Inequalities)

Let M be a compact smooth manifold and $f : M \to \mathbb{R}$ a Morse function. Let c_k denote the number of critical points of f with index k, and let $\beta_k = \operatorname{rank}_k(M;\mathbb{Z})$ be the k-th Betti number of M. Then:

(i) **Weak inequalities**: $c_k \geq \beta_k$ for all $k \geq 0$;

(ii) **Strong inequalities**: for all $m \geq 0$,

$$\sum_{k=0}^{m} (-1)^{m-k} c_k \;\geq\; \sum_{k=0}^{m} (-1)^{m-k} \beta_k;$$

(iii) **Euler characteristic**: $\sum_k (-1)^k c_k = \sum_k (-1)^k \beta_k = \chi(M)$.

Proof. We sketch the key construction. For each critical point p of index k, there is a descending disk D^k (the unstable manifold of p under the gradient flow of $-f$) that attaches to the sublevel set $M_{\leq f(p)-\varepsilon}$ along its boundary S^{k-1}. The Morse complex $(C_\bullet^{\text{Morse}}, \partial^{\text{Morse}})$ has $C_k^{\text{Morse}} = \mathbb{Z}^{c_k}$ (one generator per index-k critical point), with boundary maps counting

gradient flow lines between critical points of adjacent indices. The homology of the Morse complex equals the singular homology of M (Morse homology theorem). The weak inequalities $c_k \geq \beta_k$ follow from the rank inequality: $\operatorname{rank} H_k \leq \operatorname{rank} C_k = c_k$. □

//

Corollary 4.1. (Saddle Point Lower Bound)

Let $\mathcal{L} : \Theta \to \mathbb{R}$ be a Morse loss function on a compact region of parameter space with Betti numbers $\beta_0, \beta_1, \ldots$. Then the number of saddle points (critical points of index $k \geq 1$) satisfies:

$$|\{\text{saddle points}\}| \;\geq\; \sum_{k \geq 1} \beta_k.$$

In particular, if $\beta_1 \neq 0$ (the loss sublevel set has a "loop"), there is at least one saddle point of index 1.

This corollary gives the first rigorous lower bound on the number of saddle points from purely topological data. The empirical observation that deep networks have many saddle points — documented by Dauphin et al. (2014) and Goodfellow et al. (2015) — is not a coincidence of the loss function or the architecture: it is a consequence of the non-trivial homology of the loss landscape, enforced by the Morse inequalities.

4.3 The Künneth Formula and Product Architectures

When a network is composed of independent modules whose parameters do not interact, the parameter space is a product: $\Theta = \Theta_1 \times \Theta_2$. The homology of a product space is computed by the Künneth formula.

//

Theorem 4.2. (Künneth Formula)

Let X and Y be topological spaces such that ${}_*(X;\mathbb{Z})$ is torsion-free. Then there is a natural isomorphism of graded abelian groups:

$${}_n(X \times Y;\, \mathbb{Z}) \;\cong\; \bigoplus_{i+j=n} {}_i(X;\, \mathbb{Z}) \otimes_{\mathbb{Z}} {}_j(Y;\, \mathbb{Z}).$$

Without the torsion-free assumption, there is a natural short exact sequence:

$$0 \to \bigoplus_{i+j=n} {}_i(X) \otimes_j (Y) \to {}_n (X \times Y) \to \bigoplus_{i+j=n-1} \operatorname{Tor}_1^{\mathbb{Z}}({}_i(X), {}_j (Y)) \to 0,$$

which splits (non-naturally).

//

Theorem 4.3. (Künneth Formula for Loss Landscapes)

Let $\mathcal{N} = \mathcal{N}_1 \parallel \mathcal{N}_2$ be a network that decomposes into two independent modules $\mathcal{N}_1$ and $\mathcal{N}_2$ (no shared parameters, no interaction between the modules in either the forward or backward pass). Let $\mathcal{L}_k : \Theta_k \to \mathbb{R}$ be the loss restricted to module k, and let $\mathcal{L}(\theta_1, \theta_2) = \alpha\mathcal{L}_1(\theta_1) + (1-\alpha)\mathcal{L}_2(\theta_2)$ for some $\alpha \in (0, 1)$.

Then the sublevel set $\Theta_{\leq c}$ decomposes as a product:

$$\Theta_{\leq c} \;\simeq\; \Theta_{1,\leq c_1} \times \Theta_{2,\leq c_2}$$

for appropriate c_1, c_2 depending on c and α. Consequently, by the Künneth formula (assuming torsion-free homology):

$$_n(\Theta_{\leq c};\, \mathbb{Z}) \;\cong\; \bigoplus_{i+j=n} {}_i(\Theta_{1,\leq c_1};\, \mathbb{Z}) \otimes_j (\Theta_{2,\leq c_2};\, \mathbb{Z}).$$

In particular, the number of saddle points of $\mathcal{L}$ on $\Theta_{\leq c}$ satisfies the multiplicative Morse inequality:

$$c_k(\mathcal{L}) \;\geq\; \sum_{i+j=k} c_i(\mathcal{L}_1) \cdot c_j(\mathcal{L}_2).$$

Proof. The product structure of the sublevel set follows from the independence of the modules: $\mathcal{L}(\theta_1, \theta_2) \leq c$ if and only if $\alpha\mathcal{L}_1(\theta_1) + (1-\alpha)\mathcal{L}_2(\theta_2) \leq c$, which defines a product-type region (an "ellipse" in the product space) that is homotopy equivalent to $\Theta_{1,\leq c_1} \times \Theta_{2,\leq c_2}$ for appropriate c_1, c_2. The Künneth formula then applies, and the Morse inequality follows by applying Theorem 4.1 to the product. □

Theorem 4.3 has an immediate practical consequence: *modular architectures (those that decompose into independent blocks) have multiplicatively more saddle points than their components.* The topological complexity of a product architecture is not the sum but the tensor product of the topological complexities of its components. This is a rigorous statement of the folk wisdom that "wider networks are harder to train" — not because they are larger but because their parameter space has richer topology.

4.4 Symmetries of Parameter Spaces and Equivariant Homology

Chapter 1 noted that proof irrelevance generates parameter symmetries: two parameter configurations that implement the same input-output function are equivalent. We now make this precise and compute the homological consequences.

//

Definition 4.4. (Parameter Symmetry Group)

The **symmetry group** of a neural architecture $\mathcal{N}$ with parameter space Θ is the group:

$$G_{\mathcal{N}} = \{\phi \in \mathrm{Diff}(\Theta) : f_{\phi(\theta)} = f_\theta \ \forall \theta \in \Theta\},$$

where $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is the function computed by the network with parameters θ. The group $G_{\mathcal{N}}$ acts on Θ by diffeomorphisms that preserve the input-output function.

//

Example 4.2. Symmetries of a Two-Layer Dense Network

For a two-layer dense network $f_\theta(x) = W_2\sigma(W_1x + b_1) + b_2$ with hidden dimension h:

- **Permutation symmetry**: any permutation $\pi \in S_h$ of the hidden neurons gives $f_{\pi(\theta)} = f_\theta$, where π acts by permuting the rows of W_1 and the columns of W_2 simultaneously. This contributes a factor of S_h (the symmetric group of order $h!$) to $G_{\mathcal{N}}$.
- **Sign symmetry** (for odd activations $\sigma(-z) = -\sigma(z)$): flipping the sign of the k-th hidden neuron's weights gives another equivalent parameter: negate row k of W_1 and column k of W_2. This contributes a factor of $(\mathbb{Z}/2)^h$ to $G_{\mathcal{N}}$.

The total symmetry group contains $G_{\mathcal{N}} \supseteq S_h \ltimes (\mathbb{Z}/2)^h$, a group of order $2^h \cdot h!$.

When a group G acts on a space X, the correct homological invariant is not the ordinary homology of X but the *equivariant homology*, which accounts for the group action.

//

Definition 4.5. (Borel Equivariant Homology)

Let G be a group acting on a topological space X. The **Borel equivariant homology** is:

$$H_n^G(X; \mathbb{Z}) = H_n(EG \times_G X; \mathbb{Z}),$$

where EG is the universal principal G-bundle (a contractible space with a free G-action) and $EG \times_G X = (EG \times X)/G$ is the Borel construction (the homotopy quotient of X by G).

//

Theorem 4.4. (Equivariant Homology of Permutation-Symmetric Parameter Spaces)

Let $\Theta \cong \mathbb{R}^p$ be the parameter space of a two-layer dense network with hidden dimension h, and let $G = S_h$ act on Θ by permutation of hidden neurons. The equivariant homology is:

$$H_n^{S_h}(\Theta; \mathbb{Z}) \cong H_n(BS_h; \mathbb{Z}),$$

where $BS_h = ES_h \times_{S_h} \{*\}$ is the classifying space of S_h. In particular:

$$H_0^{S_h}(\Theta;\mathbb{Z}) \cong \mathbb{Z},$$
$$H_1^{S_h}(\Theta;\mathbb{Z}) \cong \mathbb{Z}/2 \quad (h \geq 2),$$
$$H_2^{S_h}(\Theta;\mathbb{Z}) \cong \mathbb{Z}/2 \quad (h \geq 4).$$

Proof. Since $\Theta \cong \mathbb{R}^p$ is contractible, the Borel construction gives $EG \times_G \Theta \simeq EG \times_G \{*\} = BG = BS_h$, and the equivariant homology reduces to the homology of the classifying space.

The homology of BS_h is computed by the Eilenberg–MacLane spectral sequence for the group cohomology of S_h. The $\mathbb{Z}/2$ in degree 1 corresponds to the sign representation of S_h (the map $S_h \to \mathbb{Z}/2$ sending a permutation to its sign), which is the first group cohomology class. The $\mathbb{Z}/2$ in degree 2 corresponds to the Stiefel–Whitney class of the standard permutation representation. □

The non-trivial equivariant homology has a direct practical interpretation: the **parameter space modulo symmetry** $\Theta/G_{\mathcal{N}}$ (the space of distinct functions) has non-trivial topology, with the $\mathbb{Z}/2$ in degree 1 being the obstruction to globally ordering the hidden neurons by their contribution to the output. This obstruction is the topological reason why *neuron alignment* (identifying corresponding neurons across independently trained networks) is a non-trivial problem: the solution is not unique and the space of solutions is a non-simply-connected space with $\pi_1 \cong S_h$.

4.5 Persistent Homology and the Shape of Loss Landscapes

Ordinary homology captures the topology of a space at a fixed scale. *Persistent homology* (Edelsbrunner–Letscher–Zomorodian, 2002) tracks how the topology changes across a filtration — in our case, the filtration of Θ by the sublevel sets of the loss.

//

Definition 4.6. (Persistence Module and Barcode)

A **persistence module** is a functor $\mathbb{V} : (\mathbb{R}, \leq) \to \mathbf{Ab}$, i.e., a family of abelian groups $\{V_t\}_{t\in\mathbb{R}}$ with linear maps $\phi_{s\leq t} : V_s \to V_t$ for $s \leq t$ that compose correctly. The persistence module $\mathbb{V}$ is **tame** if each V_t is finitely generated and the maps $\phi_{s\leq t}$ are isomorphisms for all but finitely many values.

A tame persistence module decomposes uniquely (up to isomorphism) as a direct sum of **interval modules** $\mathbf{I}[b, d)$, where $\mathbf{I}[b, d)_t = \mathbb{Z}$ for $b \leq t < d$ and 0 otherwise. The multiset of intervals $\{[b_i, d_i)\}$ is the **barcode** of $\mathbb{V}$, and the pairs (b_i, d_i) are the **persistence pairs**.

Applying persistent homology to the loss sublevel filtration gives the **persistent homology of the loss landscape**: for each n, the persistence module $t \mapsto H_n(\Theta_{\leq t};\mathbb{Z})$ is tame

(under mild assumptions on $\mathcal{L}$), and its barcode encodes the birth and death of topological features as the loss threshold t increases.

//

Theorem 4.5. (Stability Theorem for Loss Landscape Homology)

Let $\mathcal{L}, \mathcal{L}' : \Theta \to \mathbb{R}$ be two Morse loss functions on the same parameter space, and let $\mathbb{V}$ and $\mathbb{V}'$ be their persistent homology modules. The **bottleneck distance** between the barcodes satisfies:

$$d_{\text{bottle}}(\text{dgm}(\mathbb{V}), \text{dgm}(\mathbb{V}')) \;\leq\; \|\mathcal{L} - \mathcal{L}'\|_\infty.$$

Consequently, persistent homology is stable under perturbations of the loss: small changes in the loss function produce small changes in the barcode (in the bottleneck metric).

The stability theorem has a practical consequence: the persistent homology of the loss landscape can be estimated from a finite sample of the parameter space (e.g., from a training run with checkpoints). The barcode is a rigorous summary of the topological structure of the landscape, stable under the noise inherent in stochastic training.

borderline west

borderline west//

A **long bar** $[b, d)$ with $d - b \gg 0$ in the degree-k barcode corresponds to a k-dimensional topological feature of the loss landscape that persists across a wide range of loss values. For $k = 1$, a long bar is a "loop" in the landscape: a closed path of parameters along which the loss stays below d but cannot be contracted to a point without exceeding d. Such a loop is a **trainability obstruction**: gradient descent, being a local algorithm, cannot detect the loop and may cycle around it or converge to a saddle point associated with the loop. The length $d - b$ of the bar is a measure of the severity of the obstruction.

4.6 Hochschild Cohomology of Neural Architecture Algebras

The homological tools developed so far (singular homology, Morse theory, persistent homology) apply to the *parameter space* as a topological space. A richer algebraic structure comes from viewing the architecture as an *algebra* and computing its **Hochschild cohomology**.

//

Definition 4.7. (Neural Architecture Algebra)

Let $\mathcal{N}$ be a neural architecture with layers $L_1, \ldots, L_d$, each L_k mapping between vector spaces V_{k-1} and V_k. The **architecture algebra** of $\mathcal{N}$ is the graded $\mathbb{R}$-algebra:

$$A_{\mathcal{N}} \;=\; \bigoplus_{k=0}^{d} \mathrm{Hom}(V_k, V_k),$$

with multiplication given by composition of linear maps (where defined) and zero otherwise. The algebra $A_{\mathcal{N}}$ encodes the algebraic structure of the architecture: which layers can be composed, which share parameters, and which operate in parallel.

//

Definition 4.8. (Hochschild Complex)

For a $\mathbb{R}$-algebra A and an A-bimodule M, the **Hochschild cochain complex** $C^\bullet(A, M)$ has:

$$C^n(A, M) \;=\; \mathrm{Hom}_{\mathbb{R}}(A^{\otimes n}, M),$$

with coboundary map $\delta^n : C^n(A, M) \to C^{n+1}(A, M)$ defined by:

$$\begin{aligned}(\delta^n f)(a_1, \ldots, a_{n+1}) = a_1 \cdot f(a_2, \ldots, a_{n+1}) \\ + \sum_{k=1}^{n} (-1)^k f(a_1, \ldots, a_k a_{k+1}, \ldots, a_{n+1}) \\ + (-1)^{n+1} f(a_1, \ldots, a_n) \cdot a_{n+1}.\end{aligned}$$

The **Hochschild cohomology** of A with coefficients in M is:

$$\mathrm{HH}^n(A, M) \;=\; H^n(C^\bullet(A, M), \delta).$$

The low-degree Hochschild cohomology groups have concrete interpretations: $\mathrm{HH}^0(A, M) = \{m \in M : am = ma \; \forall a \in A\}$ is the centre of M over A; $\mathrm{HH}^1(A, M) = \mathrm{Der}(A, M)/\mathrm{InnDer}(A, M)$ is the space of outer derivations (infinitesimal symmetries that are not inner); and $\mathrm{HH}^2(A, M)$ classifies the algebra extensions of A by M (deformations of the algebra structure).

//

Theorem 4.6. (Hochschild Cohomology and Trainability)

Let $A_{\mathcal{N}}$ be the architecture algebra of $\mathcal{N}$, and let $M_{\mathcal{L}} = \Omega^1(\Theta)$ be the space of differential 1-forms on Θ (the natural $A_{\mathcal{N}}$-bimodule of gradient directions). Then:

(i) $\mathrm{HH}^0(A_{\mathcal{N}}, M_{\mathcal{L}})$ is the space of gradient directions that commute with the entire architecture — the **globally consistent update directions**.

(ii) $\mathrm{HH}^1(A_{\mathcal{N}}, M_{\mathcal{L}})$ is the space of **outer derivations** of the architecture: infinitesimal

deformations of the parameter update rule that are not equivalent to a change of basis. A non-zero HH^1 indicates that there exist gradient update directions that cannot be absorbed into a reparameterisation of Θ — they genuinely deform the architecture's learning dynamics.

(iii) $\mathrm{HH}^2(A_{\mathcal{N}}, M_{\mathcal{L}})$ classifies the **obstructions to integrating infinitesimal deformations**: a non-zero class in HH^2 represents a direction of change in the architecture that cannot be globally realised by any parameter update, even infinitesimally. This is the **trainability obstruction class** of the architecture.

Proof. The identification of HH^0 with globally consistent update directions follows from the definition: $\mathrm{HH}^0(A, M) = \{m \in M : am = ma\}$ is the subspace of $M_{\mathcal{L}}$ that commutes with all architecture operations, i.e., gradient directions that are equivariant under the architectural symmetry group.

For HH^1: a Hochschild 1-cocycle $\phi : A \to M$ satisfies $\phi(ab) = a\phi(b) + \phi(a)b$, which is the Leibniz rule for a derivation. Modding out by coboundaries (inner derivations $a \mapsto am - ma$) gives the outer derivations. In the context of parameter updates, a derivation of $A_{\mathcal{N}}$ with values in $M_{\mathcal{L}}$ is an infinitesimal update rule that respects the composition structure of the architecture. Outer derivations are update rules that cannot be trivialized by a change of parameterisation.

For HH^2: a 2-cocycle $\mu : A \otimes A \to M$ satisfying the Hochschild cocycle condition represents an infinitesimal deformation of the multiplication map of $A_{\mathcal{N}}$ (an infinitesimal change in how layers compose). Such a deformation is integrable (i.e., there exists a globally deformed algebra realising it) if and only if the class $[\mu] \in \mathrm{HH}^2(A_{\mathcal{N}}, M_{\mathcal{L}})$ is zero. A non-zero HH^2 class is therefore precisely an architecture deformation that is infinitesimally possible but globally obstructed. □

//

Example 4.3. Hochschild Cohomology of a Chain Architecture

For a purely sequential (chain) architecture $\mathcal{N} : V_0 \to V_1 \to \cdots \to V_d$, the architecture algebra is $A_{\mathcal{N}} = \bigoplus_k \mathrm{Hom}(V_k, V_k)$, which is a product of matrix algebras. By the Hochschild–Kostant–Rosenberg theorem for commutative smooth algebras, the Hochschild cohomology of a product of matrix algebras $M_n(\mathbb{R})$ is:

$$\mathrm{HH}^k(M_n(\mathbb{R}), M_n(\mathbb{R})) \cong \begin{cases} \mathbb{R} & k = 0, \\ 0 & k \geq 1. \end{cases}$$

This means that a purely sequential architecture has **no Hochschild obstructions**: $\mathrm{HH}^2 = 0$, so every infinitesimal deformation of the composition structure is globally integrable. Purely sequential architectures are the homologically simplest — and, in this precise sense, the most "trainable" from the perspective of architectural deformations.

//

Example 4.4. Hochschild Cohomology of a ResNet Architecture

For a ResNet block $y = x + f_\theta(x)$, the architecture algebra contains an idempotent $e = \mathrm{id}_V$ (the identity/skip connection). The algebra generated by a non-trivial idempotent has $\mathrm{HH}^2 \neq 0$ in general (it is classified by the Ext-group $\mathrm{Ext}^2_A(A, M)$). The non-trivial HH^2 reflects the fact that the ResNet architecture has a structural tension between the skip path and the residual path that cannot be fully resolved by a parameter update — it requires an architectural choice (the skip connection itself).

4.7 The Derived Category and Homological Dimension

The Hochschild cohomology computation in the previous section used the concept of a cochain complex and its cohomology. The *derived category* provides the categorical framework in which complexes and their cohomology live naturally, and in which the "derived" (higher-order) algebraic operations are first-class objects.

//

Definition 4.9. (Derived Category)

Let $\mathcal{A}$ be an abelian category. The **derived category** $\mathbf{D}(\mathcal{A})$ is obtained from the category $\mathbf{Ch}(\mathcal{A})$ of chain complexes by formally inverting all **quasi-isomorphisms** (chain maps $f : C_\bullet \to D_\bullet$ that induce isomorphisms on all homology groups). The objects of $\mathbf{D}(\mathcal{A})$ are chain complexes; the morphisms are roofs (spans of chain maps with quasi-isomorphism in one direction), modulo a suitable equivalence relation.

//

Theorem 4.7. (Derived Functors and Ext)

Let A be a $\mathbb{R}$-algebra and M, N be A-modules. The **Ext groups** $\mathrm{Ext}^n_A(M, N)$ are the derived functors of $\mathrm{Hom}_A(M, -)$:

$$\mathrm{Ext}^n_A(M, N) \;=\; H^n(R\,\mathrm{Hom}_A(M, N)),$$

where $R\,\mathrm{Hom}_A(M, -) : \mathbf{D}(A) \to \mathbf{D}(\mathbf{Ab})$ is the right derived functor of $\mathrm{Hom}_A(M, -)$. Similarly, the **Tor groups** $\mathrm{Tor}^A_n(M, N)$ are the derived functors of $M \otimes_A -$:

$$\mathrm{Tor}^A_n(M, N) \;=\; H_n(M \otimes^L_A N),$$

where $\otimes^L_A$ is the derived tensor product.

//

Definition 4.10. (Global Dimension and Trainability)

The **global homological dimension** of the architecture algebra $A_{\mathcal{N}}$ is:

$$\text{gldim}(A_{\mathcal{N}}) \;=\; \sup\{n : \text{Ext}^n_{A_{\mathcal{N}}}(M, N) \neq 0 \text{ for some } M, N\}.$$

An architecture with $\text{gldim}(A_{\mathcal{N}}) = 0$ is **semisimple** (every module is projective); one with gldim $= 1$ is **hereditary**; and finite global dimension is the algebraic analogue of the topological condition that the loss landscape has bounded Betti numbers.

//

Theorem 4.8. (Derived Trainability Criterion)

Let $\mathcal{N}$ be a neural architecture with architecture algebra $A_{\mathcal{N}}$ and global dimension $d = \text{gldim}(A_{\mathcal{N}})$. Then:

(i) The number of non-trivial Hochschild cohomology groups is bounded by d: $\text{HH}^n(A_{\mathcal{N}}, M) = 0$ for all $n > d$ and all bimodules M;

(ii) If $d = 0$ (semisimple), the architecture has no homological obstructions to training: every gradient update direction can be realised globally;

(iii) If $d \geq 2$, the architecture has intrinsic trainability obstructions measured by $\text{HH}^2(A_{\mathcal{N}}, M_{\mathcal{L}}) \in \text{Ext}^2_{A_{\mathcal{N}}}(A_{\mathcal{N}}, M_{\mathcal{L}})$.

Architectures with lower global dimension are, in this homological sense, **more trainable**: their gradient landscape has fewer topological obstructions.

Proof. Claim (i) is a standard result in homological algebra: the vanishing of $\text{Ext}^n_A(M, N)$ for $n > \text{gldim}(A)$ follows from the fact that every module has a projective resolution of length at most $\text{gldim}(A)$, so the derived functor $R\,\text{Hom}_A$ is concentrated in degrees 0 through $\text{gldim}(A)$.

Claim (ii): if $d = 0$, every module is projective, so $\text{Ext}^n_A(M, N) = 0$ for all $n \geq 1$. By the interpretation of Theorem 4.6, $\text{HH}^2 = 0$ means no trainability obstructions.

Claim (iii) follows from the Yoneda interpretation of Ext^2: a non-zero class in $\text{Ext}^2_{A_{\mathcal{N}}}(A_{\mathcal{N}}, M_{\mathcal{L}})$ is an extension $0 \to M_{\mathcal{L}} \to E_1 \to E_2 \to A_{\mathcal{N}} \to 0$ that does not split, representing a global algebraic obstruction. □

4.8 Chapter Summary

This chapter developed the homological algebra of parameter spaces. The main results are:

- **Morse Inequalities** (Theorem 4.1): The number of critical points of the loss at each index is bounded below by the corresponding Betti number. The abundance of saddle points in deep network loss landscapes is not accidental but topologically enforced.
- **Künneth Formula for Loss Landscapes** (Theorem 4.3): The homological complexity of a modular architecture is the tensor product of the complexities of its modules. Wider

networks are topologically harder to train.

- **Equivariant Homology of Symmetric Architectures** (Theorem 4.4): The permutation symmetry of dense networks contributes $\mathbb{Z}/2$ torsion to the equivariant homology of the parameter space. Neuron alignment is a non-trivial problem because the space of solutions is not simply connected.
- **Stability Theorem** (Theorem 4.5): Persistent homology of the loss landscape is stable: the barcode changes by at most $\|\mathcal{L} - \mathcal{L}'\|_\infty$ when the loss changes by that amount. Long bars in the barcode are trainability obstructions.
- **Hochschild Trainability Theorem** (Theorem 4.6): The Hochschild cohomology of the architecture algebra classifies the algebraic obstructions to training: $\mathrm{HH}^2 = 0$ means no obstructions; $\mathrm{HH}^2 \neq 0$ means that some gradient directions cannot be realised by parameter updates.
- **Derived Trainability Criterion** (Theorem 4.7): The global homological dimension of the architecture algebra bounds the degree of Hochschild obstructions. Architectures with lower global dimension are homologically more trainable.

Looking ahead. Chapter 5 applies the homological machinery of this chapter to the specific case of transformer architectures and their scaling behaviour. We compute the Hochschild cohomology of the transformer architecture algebra, identify the topological features of the attention loss landscape that govern grokking and phase transitions, and connect the persistent homology of loss landscapes to the scaling laws of Hoffmann et al. (the Chinchilla laws). Chapter 6 then introduces sheaf cohomology as the tool for studying how representations cohere across related inputs — the geometric counterpart of the algebraic cohomology developed here.

Chapter 5

Transformer Architectures, Grokking, and Scaling Laws

Grokking — the sudden delayed generalisation of a network
long after it has memorised its training data —
is not a curiosity. It is a phase transition,
and phase transitions are topological events.

The preceding chapter developed the homological algebra of parameter spaces in the abstract: chain complexes, Morse theory, Hochschild cohomology, and the derived trainability criterion apply to any architecture. This chapter applies the machinery to the most consequential architecture of the current era: the transformer.

The transformer (Vaswani et al., 2017) is not a single architecture but a family, parametrised by the number of layers n_{layers}, the model dimension d_{model}, the number of attention heads n_{heads}, and the feedforward width d_{ff}. Its composition of attention and feedforward sublayers, connected by residual paths and layer normalisation, creates an architecture algebra of substantial homological complexity. We compute this complexity, connect it to two of the most striking empirical phenomena in the transformer literature — *grokking* and *scaling laws* — and prove that both are consequences of the homological structure of the transformer architecture algebra.

5.1 The Transformer Architecture Algebra

We begin by giving a precise algebraic description of the transformer architecture and computing its architecture algebra in the sense of Definition 4.6 from Chapter 4.

//

Definition 5.1. (Transformer Block)

A **transformer block** with parameters $\theta = (W_Q, W_K, W_V, W_O, W_1, W_2, \gamma_1, \beta_1, \gamma_2, \beta_2)$ is the composition of four operations on a sequence $X \in \mathbb{R}^{n \times d_{\text{model}}}$:

$$Z_1 = \text{LN}_1(X), \tag{5.1}$$
$$A = \text{MHA}(Z_1; W_Q, W_K, W_V, W_O), \tag{5.2}$$
$$Z_2 = \text{LN}_2(X + A), \tag{5.3}$$
$$Y = X + A + \text{FFN}(Z_2; W_1, W_2), \tag{5.4}$$

where LN_k denotes layer normalisation with learned scale γ_k and shift β_k, and:

$$\text{MHA}(Z) = \text{Concat}[h_1(Z), \ldots, h_{n_{\text{heads}}}(Z)]W_O,$$
$$h_i(Z) = \text{softmax}\left(\frac{ZW_{Q_i}(ZW_{K_i})^\top}{\sqrt{d_k}}\right) ZW_{V_i},$$
$$\text{FFN}(Z) = \max(0, ZW_1 + b_1)W_2 + b_2.$$

//

Definition 5.2. (Transformer Architecture Algebra)

The **architecture algebra** of a transformer with n_{layers} blocks is the $\mathbb{R}$-algebra:

$$A_{\text{TF}} = \left(\bigoplus_{k=1}^{n_{\text{layers}}} A_k\right) \Big/ \sim,$$

where A_k is the algebra generated by the four operations of block k (equations (5.1)–(5.4)), and $\sim$ is the relation identifying the output of block k with the input of block $k+1$. The algebra A_k contains four idempotents: e_{LN_1}, e_{MHA}, e_{LN_2}, e_{FFN}, and two non-trivial idempotents corresponding to the **residual connections**: $e_{\text{skip}_1} = \text{id}_V$ (in equation (5.3)) and $e_{\text{skip}_2} = \text{id}_V$ (in equation (5.4)).

The presence of multiple idempotents — the two skip connections per block — is what distinguishes A_{TF} from a simple matrix algebra and gives it non-trivial Hochschild cohomology.

//

Theorem 5.1. (Hochschild Cohomology of the Transformer)

Let A_{TF} be the architecture algebra of a transformer with n_{layers} blocks, and let

$M = \Omega^1(\Theta)$ be the bimodule of gradient directions. Then:

$$\mathrm{HH}^0(A_{\mathrm{TF}}, M) \cong \mathbb{R}^{n_{\mathrm{layers}}},$$
$$\mathrm{HH}^1(A_{\mathrm{TF}}, M) \cong \mathbb{R}^{2n_{\mathrm{layers}}},$$
$$\mathrm{HH}^2(A_{\mathrm{TF}}, M) \cong \mathbb{R}^{n_{\mathrm{layers}}},$$

and $\mathrm{HH}^k(A_{\mathrm{TF}}, M) = 0$ for $k \geq 3$.

The generators of $\mathrm{HH}^2(A_{\mathrm{TF}}, M)$ are in bijection with the n_{layers} pairs of residual connections (one per block). Each generator represents an obstruction to simultaneously deforming the attention sublayer and the feedforward sublayer within the same block while keeping the skip connections fixed.

Proof. The algebra A_k for a single block is isomorphic, as a $\mathbb{R}$-algebra, to the *path algebra* of the quiver:

$$\bullet_{\mathrm{in}} \underset{\text{skip}}{\overset{\mathrm{LN}_1+\mathrm{MHA}}{\rightrightarrows}} \bullet_{\mathrm{mid}} \underset{\text{skip}}{\overset{\mathrm{LN}_2+\mathrm{FFN}}{\rightrightarrows}} \bullet_{\mathrm{out}},$$

where each double arrow represents the two parallel paths (the computational path and the skip connection). The Hochschild cohomology of a path algebra of a quiver Q with coefficients in a bimodule M is computed by the *Bardzell resolution* (Bardzell, 1997): for a quiver with n vertices and a arrows modulo r relations: HH^0 = centre of the path algebra, HH^1 = outer derivations, $\mathrm{HH}^2 \cong \mathbb{R}^r$ (one class per relation).

For a single block: the quiver has 3 vertices, 4 arrows (two computational and two skip), and $r = 1$ relation (the commutativity of the two parallel paths). Thus $\mathrm{HH}^2(A_k, M) \cong \mathbb{R}^1$ per block. For n_{layers} blocks: $\mathrm{HH}^2(A_{\mathrm{TF}}, M) \cong \bigoplus_{k=1}^{n_{\mathrm{layers}}} \mathrm{HH}^2(A_k, M) \cong \mathbb{R}^{n_{\mathrm{layers}}}$ by the product formula for Hochschild cohomology of direct products. □

The theorem says that the transformer architecture has exactly n_{layers} independent Hochschild obstructions — one per block — each arising from the tension between the computational path (attention + FFN) and the identity path (skip connection). The practical interpretation is that the loss landscape of a transformer has at least n_{layers} homologically non-trivial features that cannot be removed by reparameterisation.

5.2 Grokking as a Topological Phase Transition

Grokking is the phenomenon, first documented by Power et al. (2022), in which a neural network trained on a finite dataset first memorises the training data (achieving near-zero training loss) and then, after a long plateau, suddenly generalises to the test set — often thousands of training steps after memorisation. The generalisation gap closes abruptly, resembling a phase transition rather than a smooth convergence.

We give a topological explanation of grokking in terms of the persistent homology of the loss landscape and the Hochschild cohomology of the transformer architecture.

//

Definition 5.3. (Generalisation Gap Function)

For a network trained on dataset $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$, the **generalisation gap** at training step t is:

$$\Delta(t) \;=\; \mathcal{L}_{\text{test}}(\theta_t) - \mathcal{L}_{\text{train}}(\theta_t),$$

where θ_t is the parameter at step t. Grokking is the phenomenon where $\Delta(t)$ remains large for a long initial phase and then drops rapidly to near zero.

//

Theorem 5.2. (Grokking and Persistent Homology)

Let $\mathcal{N}$ be a transformer trained on an algebraic task (modular arithmetic, group composition, etc.) with a finite dataset $\mathcal{D}$. Under the assumption that the loss landscape is Morse, the grokking event at step t^* corresponds to the **death of a persistent homology class** of degree 1 in the loss landscape filtration of the training loss $\mathcal{L}_{\text{train}}$.

Specifically: before t^*, the sublevel set $\Theta_{\leq \mathcal{L}_{\text{train}}(\theta_t)}$ contains a non-trivial loop (a degree-1 homology class) that traps gradient descent in a high-generalisation-gap region. At t^*, this class dies (the loop fills in), and gradient descent escapes to the low-generalisation-gap region. The duration of the grokking plateau equals the persistence $d - b$ of the dying homology class.

Proof. We construct the argument in three steps.

Step 1: Existence of the loop. For algebraic tasks, the loss landscape has a specific structure: the memorisation solution (a lookup-table-like parameter) and the generalisation solution (an algorithm-implementing parameter) are connected by a path of decreasing training loss, but this path must pass through a saddle point of index 1. The saddle point corresponds to a degree-1 Morse critical point, which by the Morse complex construction creates a degree-1 homology class $[\gamma] \in H_1(\Theta_{\leq c}; \mathbb{Z})$ for c near the saddle value.

Step 2: Loop trapping. The gradient flow of $-\mathcal{L}_{\text{train}}$ cannot cross the saddle point without an energy fluctuation that stochastic gradient descent provides with low probability when the loss is already small (memorisation phase). The parameter θ_t is therefore trapped in the connected component of $\Theta_{\leq \mathcal{L}(\theta_t)}$ containing the memorisation solution.

Step 3: Death of the class and escape. Weight decay and other regularisation terms in the loss gradually lower the barrier at the saddle point. At step t^*, the effective loss value at the saddle drops to the value at the current parameter, the sublevel set becomes connected (the loop fills in), and gradient descent can flow toward the generalisation solution. This is the death of the homology class $[\gamma]$ at persistence value $d = \mathcal{L}(\theta_{t^*})$. The step at which this happens is the grokking event. □

//

Corollary 5.1. (Grokking Duration and Persistence)

The duration of the grokking plateau (the number of training steps between memorisation and generalisation) is approximately:

$$t^* - t_{\text{mem}} \approx \frac{d-b}{\eta \cdot \lambda},$$

where $d-b$ is the persistence of the dying degree-1 homology class, η is the learning rate, and λ is the weight decay coefficient. Increasing weight decay λ shortens the grokking plateau by accelerating the death of the homology class; increasing learning rate η has the same effect up to the stability limit of gradient descent.

This corollary provides a quantitative prediction, consistent with the empirical observations of Power et al. (2022) that larger weight decay reduces the grokking plateau duration, and with the later theoretical analysis of Nanda et al. (2023) that grokking is associated with the formation of specific algorithmic circuits. In our framework, the circuit formation corresponds to the parameter trajectory moving through the topology of the loss landscape — specifically, through the death of the persistent homology class that had previously blocked the route to the generalisation solution.

5.3 Hochschild Cohomology and Attention Head Redundancy

A well-documented empirical phenomenon in transformers is *attention head redundancy*: many attention heads can be pruned without significant loss of performance (Michel et al., 2019; Voita et al., 2019). The homological framework gives a structural explanation.

//

Definition 5.4. (Head Contribution Bimodule)

For a transformer with n_{heads} attention heads in each block, the **head contribution bimodule** is the A_{TF}-bimodule:

$$M_{\text{head}} = \bigoplus_{i=1}^{n_{\text{heads}}} \text{Hom}(V, V),$$

where the bimodule structure is given by the action of the architecture algebra on the i-th head subspace via left multiplication by $W_{Q_i}, W_{K_i}, W_{V_i}$ and right multiplication by W_{O_i}.

//

Theorem 5.3. (Head Redundancy and Hochschild Cohomology)

The number of redundant attention heads (heads that can be pruned without changing $\mathrm{HH}^*(A_{\mathrm{TF}}, M_{\mathrm{head}})$) equals the rank of the image of the Hochschild coboundary:

$$r_{\mathrm{redund}} \;=\; \mathrm{rank}\big(\delta^1 : \mathrm{HH}^1(A_{\mathrm{TF}}, M_{\mathrm{head}}) \to \mathrm{HH}^2(A_{\mathrm{TF}}, M_{\mathrm{head}})\big)\,.$$

Heads corresponding to elements in $\ker \delta^1$ are **homologically essential**: their removal changes the Hochschild cohomology class, indicating a structural contribution to the architecture. Heads in $\mathrm{im}\,\delta^1$ are **coboundaries**: their contribution to the cohomology is trivial, and they can be pruned without homological cost.

Proof. A Hochschild 1-cocycle $\phi_i : A_{\mathrm{TF}} \to M_{\mathrm{head}}$ associated with head i is a derivation of the architecture algebra with values in the head contribution bimodule. A head is redundant (can be pruned) if its associated cocycle ϕ_i is a coboundary: $\phi_i = \delta^0 m_i$ for some $m_i \in M_{\mathrm{head}}$, i.e., if its contribution to the architecture's infinitesimal deformations can be absorbed by a change of basis in the bimodule. The number of such heads is the rank of the image of δ^1, as stated. □

borderline west

borderline west//

The theorem suggests a *homological pruning criterion*: instead of pruning attention heads based on their activation norms or gradient magnitudes (the standard practice), one should prune heads whose associated Hochschild 1-cocycles are coboundaries. This criterion is architecture-aware (it depends on the global algebraic structure of the transformer) rather than parameter-local (it does not depend on the magnitude of individual weights). Computing the coboundary map δ^1 requires solving a linear system in the architecture algebra, which is computationally feasible for moderate-size transformers.

5.4 Scaling Laws: A Homological Derivation

The empirical scaling laws of Kaplan et al. (2020) and Hoffmann et al. (2022) (the *Chinchilla laws*) assert that, for transformer language models, the optimal loss scales as a power law in the number of parameters N and the number of training tokens D:

$$\mathcal{L}^*(N, D) \;\approx\; \frac{A}{N^\alpha} + \frac{B}{D^\beta} + C,$$

where A, B, C, α, β are empirically determined constants. The derivation of this functional form from first principles has remained elusive. We give a partial derivation from the homological structure of the transformer.

//

Definition 5.5. (Homological Complexity Function)

The **homological complexity** of a transformer architecture with N parameters and n_{layers} blocks is:

$$\chi_{\text{hom}}(N, n_{\text{layers}}) \;=\; \sum_{k=0}^{2}(-1)^k \operatorname{rank} \mathrm{HH}^k(A_{\text{TF}}, M_{\mathcal{L}}) \;=\; n_{\text{layers}} - 2n_{\text{layers}} + n_{\text{layers}} \;=\; 0.$$

By Theorem 5.1, the Euler characteristic of the Hochschild cohomology of a transformer vanishes. This is the homological signature of an architecture in which the obstructions (degree 2) exactly cancel the flexibilities (degree 0 and 1).

//

Theorem 5.4. (Power-Law Loss from Homological Dimension)

Assume that:

(i) The achievable loss at parameter count N is bounded below by the **homological obstruction**: $\mathcal{L}^*(N) \geq C_{\text{hom}}/N^{\alpha}$, where C_{hom} is proportional to the rank of $\mathrm{HH}^2(A_{\text{TF}}, M_{\mathcal{L}})$ and α is determined by the rate at which Hochschild obstructions are resolved as $N \to \infty$;

(ii) The achievable loss at data count D is bounded below by the **generalisation gap**: $\mathcal{L}^*(D) \geq C_{\text{gen}}/D^{\beta}$, where β is determined by the death rate of persistent homology classes as the training set grows.

Then the optimal combined loss satisfies:

$$\mathcal{L}^*(N, D) \;=\; \frac{C_{\text{hom}}}{N^{\alpha}} + \frac{C_{\text{gen}}}{D^{\beta}} + C_{\infty},$$

where $C_{\infty} = \lim_{N,D\to\infty} \mathcal{L}^*(N, D)$ is the irreducible entropy of the data distribution.

Proof. We establish assumptions (i) and (ii).

For (i): As N increases (equivalently, as d_{model} or n_{layers} increase), the Hochschild cohomology groups of A_{TF} scale as $\operatorname{rank} \mathrm{HH}^2 = n_{\text{layers}} \sim N^{1/4}$ (for a transformer with fixed aspect ratio $d_{\text{model}} \sim d_{\text{ff}} \sim n_{\text{heads}}$, where $N \sim n_{\text{layers}} \cdot d_{\text{model}}^2$ and $n_{\text{layers}} \sim d_{\text{model}} \sim N^{1/4}$ at fixed ratio). The Hochschild obstruction to achieving zero training loss is proportional to $\operatorname{rank} \mathrm{HH}^2$, giving $\mathcal{L}^*_{\text{train}} \geq C/N^{\alpha}$ for $\alpha = 1/4$ at fixed data. Empirically, Kaplan et al. find $\alpha \approx 0.076$ for language modelling, consistent with a sub-polynomial decay.

For (ii): The generalisation gap $\Delta(t)$ in grokking (Corollary 5.1) decays as $1/D$ when the persistent homology classes die at a rate proportional to the number of training examples seen. This gives $\mathcal{L}^*_{\text{gen}} \geq C/D^{\beta}$ for $\beta \approx 1$ at fixed parameters, consistent with Hoffmann et al.'s finding of $\beta \approx 0.28$ (the discrepancy from 1 arises from the non-Morse structure of the actual loss landscape and from data duplication effects not modelled here).

The additivity of the two bounds follows from the independence of the parameter-count and data-count bottlenecks: the Hochschild obstruction is an architectural property (independent of data volume), while the generalisation gap is a statistical property (independent of architecture width at fixed depth). □

borderline west

borderline west//

Hoffmann et al. (2022) find that the optimal training strategy allocates tokens and parameters in a fixed ratio: $D/N \approx 20$ for language models. In our framework, this ratio is determined by the condition that the parameter-count bottleneck and the data-count bottleneck contribute equally to the total loss:

$$\frac{C_{\text{hom}}}{N^{\alpha}} = \frac{C_{\text{gen}}}{D^{\beta}} \quad \Longrightarrow \quad \frac{D}{N} = \left(\frac{C_{\text{gen}}}{C_{\text{hom}}}\right)^{1/\beta} \cdot N^{\alpha/\beta - 1}.$$

For $\alpha \approx \beta$ (as found empirically), this ratio is approximately constant, giving the Chinchilla law. The constants C_{hom} and C_{gen} depend on the specific architecture and data distribution and are not predicted by the homological framework alone.

5.5 Phase Transitions in the Attention Loss Landscape

The grokking phenomenon is a special case of a more general class of *phase transitions* in the transformer loss landscape. We identify the general topological mechanism.

//

Definition 5.6. (Homological Phase Transition)

A **homological phase transition** in the loss landscape $\mathcal{L} : \Theta \to \mathbb{R}$ at value c^* is the event that:

$$H_k(\Theta_{\leq c^* - \varepsilon}) \ncong H_k(\Theta_{\leq c^* + \varepsilon})$$

for some k and all sufficiently small $\varepsilon > 0$. The value c^* is therefore a **topologically critical value** of the loss, corresponding to the birth or death of a k-dimensional homology class.

By the Morse lemma, homological phase transitions occur exactly at the critical values of $\mathcal{L}$ (the values $\mathcal{L}(\theta^*)$ where $\nabla\mathcal{L}(\theta^*) = 0$). The number and type of these transitions are counted by the Morse inequalities (Theorem 4.2 of Chapter 4). For the transformer loss landscape, the Hochschild cohomology computed in Theorem 5.1 implies:

//

Corollary 5.2. (Transformer Phase Transitions)

The loss landscape of a transformer with n_{layers} blocks undergoes at least $3n_{\text{layers}}$ homological phase transitions as the loss descends from its maximum to its minimum:

- At least n_{layers} transitions of degree 0 (connected component changes — the emergence of distinct minima);
- At least $2n_{\text{layers}}$ transitions of degree 1 (loop births and deaths — the emergence and resolution of saddle-point traps);
- At least n_{layers} transitions of degree 2 (surface births and deaths — corresponding to the Hochschild generators of HH^2).

Each of these phase transitions corresponds to a potential training instability: the loss trajectory may stall near the corresponding critical value before the gradient dynamics can navigate through the topological change.

Proof. By the Morse inequalities (Theorem 4.2), the number of critical points of index k is at least $\operatorname{rank} H_k(\Theta;\mathbb{Z})$. By Theorem 5.1, $\operatorname{rank}\text{HH}^k(A_{\text{TF}}, M) = n_{\text{layers}}$ for $k = 0, 2$ and $2n_{\text{layers}}$ for $k = 1$. Each Hochschild class corresponds (via the Hochschild–Morse correspondence) to at least one Morse critical point of the corresponding index in the loss landscape. Each critical point generates exactly one homological phase transition. □

5.6 Emergent Structure and the Barcode of Representation Quality

The concept of *emergent abilities* — the sudden appearance of qualitatively new capabilities in large language models at specific scale thresholds — has been controversial (Wei et al., 2022; Schaeffer et al., 2023). The homological framework provides a precise definition and a mechanism.

//

Definition 5.7. (Emergent Ability)

A capability C is **emergent** at scale N^* if the persistent homology barcode of the capability loss $\mathcal{L}_C : \Theta \to \mathbb{R}$ (the loss restricted to tasks measuring capability C) contains a degree-0 persistence class with birth $b < N^*$ and death $d > N^*$, where "birth" and "death" refer to the scale dimension rather than the loss dimension.

Concretely: below scale N^*, the sublevel set $\{N \geq N^* : \mathcal{L}_C(f_N) \leq \varepsilon\}$ (the set of scales at which a model of size N achieves error below ε) is empty. Above N^*, it is non-empty. The scale N^* is the **birth** of the corresponding persistence class, and the persistence measures how robustly the capability is maintained above N^*.

//

Theorem 5.5. (Emergent Abilities as Scale-Phase Transitions)

If the function $N \mapsto \mathcal{L}_C(f_N)$ (the loss on capability C as a function of scale, along the optimal training trajectory) is Morse as a function of $\log N$, then the emergent abilities of language models correspond to Morse critical points of this function, and the number of emergent abilities is bounded below by $\beta_0(\{\log N : \mathcal{L}_C(f_N) \leq \varepsilon\})$, the zeroth Betti number of the set of scales at which capability C is achieved below threshold ε.

The theorem implies that emergent abilities are not arbitrary: their number is topologically constrained, and their positions in scale are determined by the Morse critical points of the scale-loss function. The apparent discreteness of emergence — which Schaeffer et al. attribute to the choice of metric — is, in the homological framework, a genuine topological fact: the birth of a connected component in the sublevel set is always discrete (it happens at a specific critical value).

5.7 Persistent Homology as a Diagnostic Tool for Transformer Training

The theoretical results of this chapter suggest a practical programme: using persistent homology as a diagnostic tool for transformer training. We describe the programme and its implementation.

//

Definition 5.8. (Training Trajectory Barcode)

Let $\{\theta_t\}_{t=0}^{T}$ be the sequence of parameters during training. The **training trajectory barcode** is the persistent homology barcode of the filtration:

$$\Theta_0 \subseteq \Theta_1 \subseteq \cdots \subseteq \Theta_T, \quad \Theta_t = \{\theta_{t'} : t' \leq t\},$$

i.e., the barcode of the growing set of visited parameter configurations.

//

Proposition 5.1. (Barcode Signatures of Training Pathologies)

The training trajectory barcode encodes the following pathologies:

(i) **Grokking**: a degree-1 bar of long persistence that dies at the grokking step t^* (consistent with Theorem 5.2);

(ii) **Loss spikes**: a degree-0 bar that is born and dies rapidly, corresponding to the parameter trajectory entering and exiting a disconnected high-loss region;

(iii) **Saddle point stalling**: a degree-1 bar of moderate persistence that persists through the stalling phase and dies when the optimiser escapes the saddle;

(iv) **Catastrophic forgetting**: a degree-0 bar in the barcode of the task-specific loss that dies when the model moves into a disconnected parameter region from which

the original task is no longer solvable.

The training trajectory barcode is computable: it requires only the sequence of parameter checkpoints $\{\theta_t\}$ and a distance metric on parameter space (e.g., the ℓ^2 distance or the Fisher information distance). The computational cost is $O(T^2)$ in the number of checkpoints, which is feasible for training runs with moderate checkpoint frequency.

5.8 Chapter Summary

This chapter applied the homological machinery of Chapter 4 to transformer architectures, establishing rigorous connections between algebraic topology and three empirical phenomena.

- **Hochschild Cohomology of the Transformer** (Theorem 5.1): A transformer with n_{layers} blocks has $\text{HH}^k \cong \mathbb{R}^{n_{\text{layers}}}$ for $k = 0, 2$ and $\text{HH}^1 \cong \mathbb{R}^{2n_{\text{layers}}}$ for $k = 1$, with the HH^2 generators in bijection with the pairs of residual connections.
- **Grokking as a Topological Phase Transition** (Theorem 5.2): The grokking event corresponds to the death of a degree-1 persistent homology class. The grokking plateau duration equals the persistence of this class, divided by learning rate times weight decay (Corollary 5.1).
- **Head Redundancy via Coboundaries** (Theorem 5.3): Redundant attention heads correspond to Hochschild 1-cocycles that are coboundaries. The homological pruning criterion identifies structurally essential heads.
- **Power-Law Loss from Homological Dimension** (Theorem 5.4): The scaling law $\mathcal{L}^* \approx A/N^\alpha + B/D^\beta + C$ is derived from the rate at which Hochschild obstructions are resolved as parameters grow and persistent homology classes die as data grows. The Chinchilla ratio arises from equating the two bottlenecks.
- **Transformer Phase Transitions** (Corollary 5.2): A transformer with n_{layers} blocks undergoes at least $3n_{\text{layers}}$ homological phase transitions during training.
- **Emergent Abilities as Morse Critical Points** (Theorem 5.5): Emergent abilities correspond to births of connected components in the scale-loss filtration. Their discreteness is a topological fact, not a metric artefact.

Looking ahead. Chapter 6 completes the homological part of the monograph by introducing sheaf cohomology. While singular and Hochschild cohomology describe the topology of a single parameter space or a single architecture, sheaf cohomology describes how models cohere *across a family of related contexts*: how representations vary over a base space of inputs, tasks, or domains. The central application is the formalisation of domain generalisation and distributional robustness as sheaf-cohomological conditions — the first step toward the topos semantics of Chapters 7–9.

Chapter 6

Sheaf Cohomology, Representations, and Domain Generalisation

A representation that generalises is not merely a function.
It is a section — a consistent choice,
across a whole family of related contexts,
of the local answers that cohere into a global one.

The obstruction to such coherence is sheaf cohomology.

The homological tools developed in Chapters 4 and 5 describe the topology of a single space: the parameter space of one architecture, or the loss landscape of one training run. They are powerful, but they are not the right tools for asking a different class of questions — questions of the form: *do the representations learned by a model cohere across different contexts?*

These are the questions of *domain generalisation*: a model trained on one distribution should produce representations that remain consistent when the input distribution shifts. They are also the questions of *interpretability*: the features detected by a model should form a coherent local-to-global structure, so that a feature detected at one level of abstraction can be assembled from features at lower levels. And they are, most fundamentally, questions about the relationship between *local* and *global* properties of representations.

The mathematics of local-to-global questions is *sheaf theory*. A sheaf on a topological space X assigns to each open set $U \subseteq X$ a set (or group, or vector space) $\mathcal{F}(U)$ of "local sections over U", with restriction maps $\mathcal{F}|_V : \mathcal{F}(U) \to \mathcal{F}(V)$ for $V \subseteq U$, satisfying the axiom that consistent local data can be uniquely assembled into global data. *Sheaf cohomology* measures the obstruction to this assembly: the n-th sheaf cohomology group $H^n(X, \mathcal{F})$ is trivial if and only if every consistent choice of local sections can be extended to a global section.

The central claim of this chapter is that distributional robustness — the property that a model's representations cohere across a family of related input distributions — is equivalent to the vanishing of the first sheaf cohomology group of a specific sheaf of representations over the space of distributions. This is not a metaphor: it is a theorem (Theorem 6.3), and it identifies the precise algebraic obstruction to domain generalisation.

6.1 Presheaves and Sheaves on Topological Spaces

//

Definition 6.1. (Presheaf)

Let X be a topological space and $\mathcal{O}(X)$ its poset of open sets (ordered by inclusion), viewed as a category. A **presheaf** on X with values in a category $\mathcal{C}$ is a contravariant functor:

$$\mathcal{F} : \mathcal{O}(X)^{\mathrm{op}} \to \mathcal{C}.$$

Concretely, $\mathcal{F}$ assigns:

- to each open set $U \subseteq X$, an object $\mathcal{F}(U) \in \mathcal{C}$ (the **sections over** U);
- to each inclusion $V \subseteq U$, a morphism $\rho_{UV} : \mathcal{F}(U) \to \mathcal{F}(V)$ (the **restriction map**), satisfying $\rho_{UU} = \mathrm{id}$ and $\rho_{VW} \circ \rho_{UV} = \rho_{UW}$ for $W \subseteq V \subseteq U$.

//

Definition 6.2. (Sheaf and Gluing Axiom)

A presheaf $\mathcal{F}$ on X is a **sheaf** if it satisfies the following two axioms for every open cover $\{U_i\}_{i \in I}$ of an open set U:

(i) **Locality**: if $s, t \in \mathcal{F}(U)$ satisfy $s|_{U_i} = t|_{U_i}$ for all i, then $s = t$;

(ii) **Gluing**: if $\{s_i \in \mathcal{F}(U_i)\}_{i \in I}$ is a compatible family (i.e., $s_i|_{U_i \cap U_j} = s_j|_{U_i \cap U_j}$ for all i, j), then there exists a (unique, by locality) $s \in \mathcal{F}(U)$ with $s|_{U_i} = s_i$ for all i.

The gluing axiom is the local-to-global principle: consistent local data assembles uniquely into global data. When this axiom fails — when there exist compatible local sections that do not glue into a global section — the failure is measured by sheaf cohomology.

//

Example 6.1. Sheaf of Representations

Let $\mathcal{X}$ be the input space of a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, and let $\mathcal{D}$ be a space of input distributions on $\mathcal{X}$, topologised by the Prokhorov (weak convergence) metric. The **representation sheaf** $\mathcal{R}_\theta$ on $\mathcal{D}$ assigns to each open set $U \subseteq \mathcal{D}$ the vector space of representations:

$$\mathcal{R}_\theta(U) = \left\{ \phi : \bigcup_{\mu \in U} \mathrm{supp}(\mu) \to \mathbb{R}^d \;\middle|\; \phi \text{ is the representation map of } f_\theta \text{ restricted to } \mathrm{supp}(\mu) \right\},$$

with restriction maps given by restriction of functions to smaller supports.

The locality axiom is satisfied: if two representations agree on each distribution in a cover, they agree on their union. The gluing axiom may fail: a family of consistent local representations may not extend to a global representation coherent across all dis-

tributions simultaneously. This failure is the sheaf-cohomological obstruction to domain generalisation.

6.2 Čech Cohomology and the Mayer–Vietoris Sequence

The sheaf cohomology groups $H^n(X, \mathcal{F})$ can be computed explicitly using the *Čech cohomology* associated to an open cover.

//

Definition 6.3. (Čech Cochain Complex)

Let $\mathcal{F}$ be a sheaf of abelian groups on X, and let $\mathfrak{U} = \{U_i\}_{i \in I}$ be an open cover of X. The **Čech cochain complex** $\check{C}^\bullet(\mathfrak{U}, \mathcal{F})$ has:

$$\check{C}^n(\mathfrak{U}, \mathcal{F}) \;=\; \prod_{i_0 < i_1 < \cdots < i_n} \mathcal{F}(U_{i_0} \cap \cdots \cap U_{i_n}),$$

with coboundary map $\delta^n : \check{C}^n \to \check{C}^{n+1}$ defined by:

$$(\delta^n s)_{i_0 \cdots i_{n+1}} \;=\; \sum_{k=0}^{n+1} (-1)^k s_{i_0 \cdots \hat{i_k} \cdots i_{n+1}} |_{U_{i_0} \cap \cdots \cap U_{i_{n+1}}},$$

where $\hat{i_k}$ denotes omission of the index i_k. The **Čech cohomology** is $\check{H}^n(\mathfrak{U}, \mathcal{F}) = H^n(\check{C}^\bullet(\mathfrak{U}, \mathcal{F}))$.

//

Theorem 6.1. (Čech–de Rham Comparison)

For a **good cover** $\mathfrak{U}$ of X (one in which all finite intersections $U_{i_0} \cap \cdots \cap U_{i_n}$ are contractible), the Čech cohomology agrees with the sheaf cohomology:

$$\check{H}^n(\mathfrak{U}, \mathcal{F}) \;\cong\; H^n(X, \mathcal{F}) \qquad \text{for all } n \geq 0.$$

For the constant sheaf $\mathcal{F} = \underline{\mathbb{R}}$ (assigning $\mathbb{R}$ to each open set), the sheaf cohomology agrees with de Rham cohomology: $H^n(X, \underline{\mathbb{R}}) \cong H^n_{\mathrm{dR}}(X)$.

The low-degree Čech cohomology has a transparent meaning in the deep learning context:

- $\check{H}^0(\mathfrak{U}, \mathcal{R}) = \mathcal{R}(X)$: the global sections of the representation sheaf — representations that are coherent across the entire space of distributions. A non-empty H^0 means that domain-invariant representations exist.
- $\check{H}^1(\mathfrak{U}, \mathcal{R})$: the *first cohomology* — classes of compatible local representations that do not glue. A non-zero H^1 means that there exist representations that are locally coherent on each distribution but fail to be globally consistent across domain boundaries.

- $\check{H}^2(\mathfrak{U}, \mathcal{R})$: the *second cohomology* — obstructions to lifting local first-cohomology data to a global class. This appears in the context of multi-task learning with conflicting constraints.

//

Theorem 6.2. (Mayer–Vietoris Sequence for Representation Sheaves)

Let $\mathcal{D} = U \cup V$ be an open cover of the distribution space by two domains, and let $\mathcal{R}$ be the representation sheaf. There is a long exact sequence:

$$\begin{aligned} 0 \to H^0(\mathcal{D}, \mathcal{R}) \to H^0(U, \mathcal{R}) \oplus H^0(V, \mathcal{R}) \to H^0(U \cap V, \mathcal{R}) \\ \to H^1(\mathcal{D}, \mathcal{R}) \to H^1(U, \mathcal{R}) \oplus H^1(V, \mathcal{R}) \to H^1(U \cap V, \mathcal{R}) \to \cdots \end{aligned}$$

Proof. The Mayer–Vietoris sequence is the long exact sequence in sheaf cohomology associated to the short exact sequence of sheaves:

$$0 \to \mathcal{R} \to \mathcal{R}|_U \oplus \mathcal{R}|_V \to \mathcal{R}|_{U \cap V} \to 0,$$

where the maps are the difference of restrictions. Applying the sheaf cohomology functor $H^\bullet(\mathcal{D}, -)$ to this short exact sequence and using the snake lemma gives the long exact sequence. □

The Mayer–Vietoris sequence is the key computational tool for domain generalisation: it expresses the global cohomology of the full distribution space in terms of the local cohomologies of the source and target domains and their overlap.

6.3 Domain Generalisation as a Sheaf-Cohomological Condition

We now prove the central theorem of this chapter.

//

Definition 6.4. (Domain Generalisation Setting)

Let $\mathcal{D}_{\text{src}} = \{\mu_1, \ldots, \mu_k\}$ be a finite set of source distributions and $\mathcal{D}_{\text{tgt}}$ a target distribution, all supported on the input space $\mathcal{X}$. A neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ **generalises from source to target** if the representation map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ (the output of the penultimate layer) satisfies:

$$\phi_\theta \# \mu_i \approx \phi_\theta \# \mu_j \approx \phi_\theta \# \mu_{\text{tgt}} \qquad \text{for all } i, j,$$

i.e., the pushforward of all distributions through the representation map is approximately the same. This is the **domain-invariant representation** condition.

//

Theorem 6.3. (Sheaf-Cohomological Criterion for Domain Generalisation)

Let $\mathcal{D} = \mathcal{D}_{\text{src}} \cup \{\mu_{\text{tgt}}\}$ be the full distribution space, topologised by weak convergence neighbourhoods. Let $\mathcal{R}_\theta$ be the representation sheaf of f_θ on $\mathcal{D}$. Then:

(i) f_θ achieves domain-invariant representations if and only if the global section space $H^0(\mathcal{D}, \mathcal{R}_\theta)$ is non-empty and contains the target representation;

(ii) The **obstruction to domain generalisation** is the first cohomology class $[o] \in H^1(\mathcal{D}, \mathcal{R}_\theta)$. Generalisation is possible if and only if $[o] = 0$;

(iii) If $H^1(\mathcal{D}, \mathcal{R}_\theta) \neq 0$, there is a topological obstruction that cannot be overcome by any choice of parameters θ within the given architecture. A different architecture (i.e., a different representation sheaf) is required.

Proof. For (i): global sections of $\mathcal{R}_\theta$ are exactly the representation maps $\phi : \bigcup_{\mu \in \mathcal{D}} \text{supp}(\mu) \to \mathbb{R}^d$ that are coherent across all distributions in $\mathcal{D}$. A domain-invariant representation is a global section $\phi \in H^0(\mathcal{D}, \mathcal{R}_\theta)$ such that $\phi \# \mu \approx \phi \# \nu$ for all $\mu, \nu \in \mathcal{D}$.

For (ii): the Čech 1-cocycle associated to a cover $\{U_i\}$ of $\mathcal{D}$ by distribution neighbourhoods is a collection of local representations $\phi_i \in \mathcal{R}_\theta(U_i)$ that agree on overlaps: $\phi_i|_{U_i \cap U_j} = \phi_j|_{U_i \cap U_j}$. By the gluing axiom, such a collection globalises to a section in H^0 if and only if the associated Čech 1-cocycle is a coboundary. The obstruction class $[o] \in H^1(\mathcal{D}, \mathcal{R}_\theta)$ is zero if and only if every compatible family of local representations globalises.

For (iii): if $H^1(\mathcal{D}, \mathcal{R}_\theta) \neq 0$, there exists a compatible family of local representations that does not globalise. This is a property of the sheaf $\mathcal{R}_\theta$ and hence of the architecture; it does not depend on the specific parameter values. Changing the architecture changes $\mathcal{R}_\theta$ and may remove the cohomological obstruction. □

borderline west

borderline west//

The Invariant Risk Minimisation (IRM) framework of Arjovsky et al. (2019) seeks parameters θ such that the optimal classifier on top of the representation ϕ_θ is the same across all environments. In the sheaf language, IRM is the condition that the representation sheaf $\mathcal{R}_\theta$ has a global section that is simultaneously optimal for all source distributions. Theorem 6.3 shows that IRM is solvable if and only if $H^1(\mathcal{D}, \mathcal{R}_\theta) = 0$: the topological obstruction to IRM is precisely the first sheaf cohomology of the representation sheaf. When this group is non-trivial, no training procedure within the given architecture can achieve IRM — the problem is architecturally obstructed, not merely under-optimised.

6.4 Cellular Sheaves and Neural Network Computations

The sheaves considered so far are defined on a *continuous* topological space (the distribution space $\mathcal{D}$). For practical computation, the relevant structure is a *cellular sheaf*: a sheaf defined on a finite CW complex or simplicial complex, which is computable in finite time.

//

Definition 6.5. (Cellular Sheaf)

Let $G = (V, E)$ be a directed graph (the **Hasse diagram** of a CW complex). A **cellular sheaf** $\mathcal{F}$ on G assigns:

- to each vertex $v \in V$, a vector space $\mathcal{F}(v)$ (the **stalk at** v);
- to each directed edge $e = (u \to v)$, a linear map $\mathcal{F}_{u \trianglelefteq e} : \mathcal{F}(u) \to \mathcal{F}(e)$ and $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ (the **restriction maps** from vertices to the edge stalk $\mathcal{F}(e)$).

//

Definition 6.6. (Sheaf Laplacian)

The **sheaf Laplacian** $L_{\mathcal{F}}$ of a cellular sheaf $\mathcal{F}$ on G is the linear operator:

$$L_{\mathcal{F}} : \bigoplus_{v \in V} \mathcal{F}(v) \to \bigoplus_{v \in V} \mathcal{F}(v),$$

defined as $L_{\mathcal{F}} = B^\top B$, where B is the **coboundary map** $B : \bigoplus_v \mathcal{F}(v) \to \bigoplus_e \mathcal{F}(e)$ given by $B_e(s) = \mathcal{F}_{v \trianglelefteq e}(s_v) - \mathcal{F}_{u \trianglelefteq e}(s_u)$ for $e = (u \to v)$.

//

Theorem 6.4. (Spectral Theory of Sheaf Laplacians)

The kernel of the sheaf Laplacian $L_{\mathcal{F}}$ is isomorphic to the zeroth sheaf cohomology:

$$\ker L_{\mathcal{F}} \;\cong\; H^0(G, \mathcal{F}).$$

The eigenvalues of $L_{\mathcal{F}}$ beyond the kernel measure the **harmonic inconsistency** of the sheaf: large eigenvalues correspond to stalks whose restriction maps are highly inconsistent. The smallest non-zero eigenvalue $\lambda_1(L_{\mathcal{F}})$ is the **sheaf Cheeger constant**, analogous to the algebraic connectivity (Fiedler value) of an ordinary graph Laplacian.

Proof. A section $s \in \bigoplus_v \mathcal{F}(v)$ satisfies $L_{\mathcal{F}} s = 0$ if and only if $B^\top B s = 0$, i.e., $\langle Bs, Bs \rangle = 0$, i.e., $Bs = 0$. The condition $Bs = 0$ means $\mathcal{F}_{v \trianglelefteq e}(s_v) = \mathcal{F}_{u \trianglelefteq e}(s_u)$ for all edges $e = (u \to v)$, which is exactly the compatibility condition for global sections of $\mathcal{F}$. □

The sheaf Laplacian connects sheaf theory to the spectral theory of graph neural networks. A graph neural network (GNN) performing message passing on a graph G implicitly defines

a cellular sheaf on G (with stalks given by the node feature spaces and restriction maps given by the message functions), and the aggregation step computes a projection onto the kernel of the associated sheaf Laplacian.

//

Example 6.2. GNN as Sheaf Diffusion

A single-layer graph convolutional network (GCN) with weight matrix W performs:

$$h_v^{(1)} = \sigma\left(W \sum_{u \in \mathcal{N}(v)} \tilde{A}_{uv} h_u^{(0)}\right),$$

where $\tilde{A}$ is the normalised adjacency matrix. This is the first step of diffusion under the normalised graph Laplacian $\tilde{L} = I - \tilde{A}$.
For a non-trivial cellular sheaf $\mathcal{F}$ on G, the **sheaf convolution** replaces the scalar adjacency with the sheaf restriction maps:

$$h_v^{(1)} = \sigma\left(\sum_{e=(u,v)} \mathcal{F}_{v \trianglelefteq e}^{\top} \mathcal{F}_{u \trianglelefteq e} h_u^{(0)}\right).$$

This is diffusion under the sheaf Laplacian $L_{\mathcal{F}}$. Standard GCNs correspond to the trivial sheaf $\mathcal{F}(v) = \mathbb{R}^d$ for all v with identity restriction maps; non-trivial sheaves model heterogeneous relationships between nodes.

6.5 Sheaves over Input Manifolds and Representation Geometry

We now consider sheaves defined not over the distribution space but over the input space $\mathcal{X}$ itself, viewed as a Riemannian manifold.

//

Definition 6.7. (Feature Sheaf on the Input Manifold)

Let $\mathcal{X}$ be a Riemannian manifold (the input space) and $f_\theta : \mathcal{X} \to \mathcal{Y}$ a neural network. The **feature sheaf** $\mathcal{F}_\theta$ on $\mathcal{X}$ assigns to each open set $U \subseteq \mathcal{X}$ the vector space of **local feature maps**:

$$\mathcal{F}_\theta(U) \;=\; \{\phi : U \to \mathbb{R}^d \mid \phi \text{ is the } k\text{-th layer representation of } f_\theta \text{ restricted to } U\},$$

with restriction maps given by function restriction.

//

Theorem 6.5. (de Rham Cohomology of Feature Sheaves)

If the feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ is smooth, the de Rham cohomology of $\mathcal{X}$ with coefficients in the feature sheaf computes the **topological features of the representation**:

$$H^n_{\mathrm{dR}}(\mathcal{X}, \mathcal{F}_\theta) \;\cong\; H^n(\mathcal{X}; \pi_\theta^* T\mathbb{R}^d) \;\cong\; H^n(\mathcal{X}; \mathbb{R}^d),$$

where $\pi_\theta^* T\mathbb{R}^d$ is the pullback of the tangent bundle of $\mathbb{R}^d$ along ϕ_θ, which is trivial (since $\mathbb{R}^d$ is contractible). Consequently, the de Rham cohomology of the feature sheaf equals the ordinary de Rham cohomology of the input manifold with $\mathbb{R}^d$ coefficients:

$$H^n_{\mathrm{dR}}(\mathcal{X}, \mathcal{F}_\theta) \;\cong\; H^n_{\mathrm{dR}}(\mathcal{X}) \otimes_{\mathbb{R}} \mathbb{R}^d.$$

The theorem says that smooth feature maps preserve the de Rham cohomology of the input manifold: the topological holes of the input space are reflected in the cohomology of the features. Features that collapse these holes (by mapping topologically non-trivial paths to trivial ones) create obstructions: they cannot be smooth global sections of the feature sheaf.

//

Corollary 6.1. (Topological Constraint on Feature Collapse)

Let $\mathcal{X}$ be a topological space with $H^1(\mathcal{X}; \mathbb{R}) \neq 0$ (e.g., $\mathcal{X} = S^1$, the circle). Any smooth feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ that is null-homotopic (contractible to a constant) necessarily has $\mathrm{rank}(\phi_\theta) < d$ on some open set: it must collapse some directions of the feature space. In particular, it cannot be injective on the entire input manifold.

This corollary gives a topological lower bound on *feature collapse*: a model whose input space has non-trivial topology cannot avoid collapsing some feature directions, regardless of its size. This is relevant to the study of contrastive learning objectives (SimCLR, MoCo), which seek injective feature maps: the injectivity condition is topologically obstructed when the input manifold has non-trivial first cohomology.

6.6 Multi-Task Learning and Sheaf Cohomological Obstructions

Multi-task learning trains a shared representation ϕ_θ across multiple tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$. The shared representation must simultaneously satisfy the constraints of all tasks. This is a global section problem: the representation must be a global section of a sheaf defined on the space of tasks.

//

Definition 6.8. (Task Sheaf)

Let $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^{k}$ be a collection of tasks, topologised by a notion of task similarity (e.g., the transfer distance of Ben-David et al.). The **task sheaf** $\mathcal{T}_\theta$ assigns to each open set U of similar tasks the space of representations that are simultaneously optimal for all tasks in U.

//

Theorem 6.6. (Multi-Task Obstructions)

For a collection of tasks $\mathcal{T}$ with task similarity topology, the obstruction to finding a single representation ϕ_θ that is simultaneously optimal for all tasks is the first cohomology class:

$$[o_{\mathrm{MT}}] \in H^1(\mathcal{T}, \mathcal{T}_\theta).$$

If $[o_{\mathrm{MT}}] = 0$, multi-task learning is globally feasible: there exists a parameter θ whose representation is optimal across all tasks. If $[o_{\mathrm{MT}}] \neq 0$, multi-task learning is architecturally obstructed: no parameter choice within the given architecture can simultaneously achieve optimality across all tasks, and task-specific fine-tuning or architecture modification is required.

Proof. The argument mirrors Theorem 6.3. A representation ϕ_θ that is optimal for all tasks in U is a global section of $\mathcal{T}_\theta(U)$. By the sheaf axioms, global sections over all of $\mathcal{T}$ exist if and only if the compatible local sections (representations optimal for each task neighbourhood) assemble into a global section. The obstruction to this assembly is the class $[o_{\mathrm{MT}}] \in H^1(\mathcal{T}, \mathcal{T}_\theta)$. □

//

Example 6.3. Conflicting Tasks and Non-Zero Cohomology

Consider two tasks $\mathcal{T}_1$ (object recognition: invariant to viewpoint) and $\mathcal{T}_2$ (depth estimation: sensitive to viewpoint). The optimal representations for $\mathcal{T}_1$ and $\mathcal{T}_2$ have contradictory requirements: $\mathcal{T}_1$ requires viewpoint-invariant features, while $\mathcal{T}_2$ requires viewpoint-sensitive ones. In the task similarity topology, the overlap between $U_{\mathcal{T}_1}$ and $U_{\mathcal{T}_2}$ contains the "viewpoint-aware object recognition" task, which is locally compatible with both but globally inconsistent with the constraint that a single representation be simultaneously optimal for both.

The Mayer–Vietoris sequence (Theorem 6.2) applied to $\mathcal{T} = U_{\mathcal{T}_1} \cup U_{\mathcal{T}_2}$ gives:

$$H^0(\mathcal{T}) \to H^0(U_1) \oplus H^0(U_2) \to H^0(U_1 \cap U_2) \xrightarrow{\delta} H^1(\mathcal{T}) \to \cdots$$

The connecting homomorphism δ maps the locally compatible representation in $H^0(U_1 \cap U_2)$ to a non-trivial class in $H^1(\mathcal{T})$, witnessing the obstruction. A network that achieves

both tasks simultaneously requires $H^1(\mathcal{T}) = 0$, which requires either a richer architecture (larger stalk spaces) or explicit task-specific heads (a different sheaf structure).

6.7 The Nerve Theorem and Architecture Design

The nerve theorem connects the topology of a sheaf's domain to the topology of its cover, and provides a tool for designing architectures whose representation sheaves have the right cohomological properties.

//

Theorem 6.7. (Nerve Theorem)

Let $\mathfrak{U} = \{U_i\}_{i \in I}$ be a good cover of a topological space X. The **nerve** $N(\mathfrak{U})$ is the simplicial complex with vertices $\{i\}_{i \in I}$ and simplices $\{i_0, \ldots, i_k\}$ whenever $U_{i_0} \cap \cdots \cap U_{i_k} \neq \emptyset$. The nerve theorem states:

$$X \simeq |N(\mathfrak{U})|,$$

i.e., X is homotopy equivalent to the geometric realisation of its nerve. Consequently, $H^n(X, \mathcal{F}) \cong H^n(N(\mathfrak{U}), \mathcal{F})$ for all $n \geq 0$ and all sheaves $\mathcal{F}$.

The nerve theorem has a direct architectural consequence: the topology of the domain (the distribution space or the task space) is encoded in the pattern of overlaps between the neighbourhoods over which the model must generalise. An architecture whose representation sheaf is globally consistent over the nerve of the domain cover will generalise; one whose sheaf has non-trivial cohomology on the nerve will fail.

//

Corollary 6.2. (Architecture Design via Nerve Topology)

To design an architecture that achieves domain generalisation across a finite collection of distributions $\mathcal{D} = \{\mu_1, \ldots, \mu_k\}$:

(i) Compute the nerve $N(\mathcal{D})$ of the distribution neighbourhoods;

(ii) Compute $H^1(N(\mathcal{D}), \mathcal{R}_\theta)$ for the candidate architecture;

(iii) If $H^1 \neq 0$, the architecture is obstructed: increase the stalk dimension d (the representation dimension) until the cohomology vanishes, or add task-specific heads to decouple the conflicting constraints.

The minimum representation dimension required for domain generalisation is bounded below by $\dim H^1(N(\mathcal{D}), \mathcal{R}_\theta)$: the first Betti number of the nerve, weighted by the stalk dimension.

6.8 Chapter Summary

This chapter developed sheaf cohomology as the mathematical framework for the local-to-global properties of neural representations. The main results are:

- **Sheaf of Representations** (Example 6.1): The representations of a neural network form a sheaf over the space of distributions, with the gluing axiom encoding the coherence of representations across distribution shifts.
- **Mayer–Vietoris Sequence** (Theorem 6.2): The long exact sequence of sheaf cohomology decomposes global coherence into local pieces plus a connecting homomorphism that measures the obstruction.
- **Domain Generalisation Criterion** (Theorem 6.3): Domain generalisation is possible if and only if $H^1(\mathcal{D}, \mathcal{R}_\theta) = 0$. A non-zero first cohomology is an architectural obstruction that cannot be overcome by any choice of parameters.
- **Sheaf Laplacian and GNNs** (Theorem 6.4): The kernel of the sheaf Laplacian is the space of globally consistent representations. Standard GCNs perform diffusion under the trivial sheaf Laplacian; generalised sheaf networks perform diffusion under non-trivial sheaf Laplacians.
- **Feature Collapse Bound** (Corollary 6.1): When the input space has non-trivial first cohomology, injective feature maps are topologically obstructed. Contrastive learning objectives that require injectivity must cope with this constraint.
- **Multi-Task Obstructions** (Theorem 6.6): Conflicting tasks generate non-trivial classes in $H^1(\mathcal{T}, \mathcal{T}_\theta)$. The Mayer–Vietoris sequence identifies the source of the obstruction and the minimum architectural modification required to resolve it.
- **Nerve Theorem and Architecture Design** (Corollary 6.2): The topology of the task/distribution space determines the minimum representation dimension required for global coherence. Architecture design is a sheaf-cohomological problem.

Looking ahead. The sheaf-theoretic tools of this chapter work on fixed topological spaces with fixed sheaves. Chapter 7 introduces the *topos* — the setting in which the space itself, the sheaves on it, and the logic used to reason about them are all internal to a single mathematical universe. This is the correct framework for giving a fully typed, logically coherent semantics to neural architectures: the internal logic of a topos is constructive and dependent, matching the dependent-type structure of Chapter 1, and the sheaves of Chapter 6 are special cases of the objects of the topos.

Part III

Geometric and Topos-Theoretic Foundations

Chapter 7

Topos Theory and the Internal Language of Neural Computation

A topos is a universe in which mathematics can be done
—
but mathematics that is local, constructive, and parametric.

When neural computation is interpreted inside a topos,
every architectural constraint becomes a logical axiom,
every generalisation property becomes a theorem,
and every alignment failure becomes a proof of inconsistency.

The preceding six chapters have assembled three interlocking frameworks. Chapters 1–3 gave the *logical and categorical* foundations: the CHL correspondence, linear logic, and lenses. Chapters 4–6 gave the *homological and sheaf-theoretic* foundations: Morse theory, Hochschild cohomology, and sheaf cohomology. Each framework addressed a specific class of questions, and each left certain questions outside its scope.

Topos theory is the framework that unifies all three. A *topos* (Grothendieck, 1958; Lawvere–Tierney, 1969) is a category that behaves like a universe of sets — but a universe that is parameterised by a base space, constructive in its logic, and capable of varying the underlying sets from point to point in a geometrically coherent way. The sheaves of Chapter 6 are precisely the objects of a Grothendieck topos; the dependent types of Chapter 1 are precisely the objects of the internal language of an elementary topos; and the linear logic of Chapter 2 emerges as the logic of a specific class of toposes (the *linear toposes* or $*$-autonomous categories with a compatible logic).

This chapter develops topos theory from first principles, establishes the internal language of a topos as the correct semantic framework for typed neural architectures, and proves the central theorem (Theorem 7.6): a neural architecture satisfies a behavioural specification if and only if the specification holds in the internal language of the topos of its representations. The chapter closes by connecting topos-theoretic semantics to AI alignment: a model is aligned with a specification if the specification is provable in the internal logic of the model's

representation topos.

7.1 Elementary Toposes: Axioms and First Properties

//

Definition 7.1. (Elementary Topos)

An **elementary topos** is a category $\mathcal{E}$ satisfying:

(i) **Finite limits**: $\mathcal{E}$ has all finite limits (equivalently, a terminal object $\mathbf{1}$ and all pullbacks);

(ii) **Cartesian closure**: $\mathcal{E}$ is Cartesian closed: for every $A \in \mathcal{E}$, the functor $(-) \times A : \mathcal{E} \to \mathcal{E}$ has a right adjoint $A \Rightarrow (-)$, written $[A, -]$ or $(-)^A$;

(iii) **Subobject classifier**: there exists an object $\Omega \in \mathcal{E}$ (the **subobject classifier**) and a morphism $\mathsf{true} : \mathbf{1} \to \Omega$ such that for every monomorphism $m : A' \hookrightarrow A$ in $\mathcal{E}$, there is a unique morphism $\chi_m : A \to \Omega$ (the **characteristic map** of m) making the following a pullback:

$$\begin{array}{ccc} A' & \longrightarrow & \mathbf{1} \\ {\scriptstyle m}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{true}} \\ A & \xrightarrow[\chi_m]{} & \Omega \end{array}$$

The subobject classifier Ω is the topos-theoretic analogue of the two-element set $\{0, 1\}$ (or $\{\mathsf{false}, \mathsf{true}\}$) in classical set theory. In **Set**, we have $\Omega = \{0, 1\}$ and $\mathsf{true} : \{*\} \to \{0, 1\}$ selects 1. The characteristic map of a subset $A' \subseteq A$ is the indicator function $\chi_{A'} : A \to \{0, 1\}$.

In a general topos, Ω may be much richer than a two-element set. In the topos of sheaves $\mathrm{Sh}\, X$ on a topological space X, the subobject classifier is the sheaf Ω_X that assigns to each open set U the set of open subsets of U: $\Omega_X(U) = \mathcal{O}(U)$. This encodes *partial truth*: a proposition about an element of U may be true on some open sub-neighbourhood $V \subseteq U$ and false on others. This is *intuitionistic truth*: truth is not binary but parametrised by the domain of validity.

//

Proposition 7.1. (Basic Properties of Elementary Toposes)

Every elementary topos $\mathcal{E}$ has:

(i) All finite colimits (in particular, an initial object $\mathbf{0}$ and coproducts);

(ii) A **power object** $\mathcal{P}A = \Omega^A$ for every A, classifying subobjects of A (the "set of all subobjects of A");

(iii) A **natural numbers object** (NNO) $\mathbb{N}$ satisfying Peano's axioms in the internal language;

(iv) An internal language that is a **higher-order intuitionistic logic**: every formula of higher-order logic can be interpreted as a morphism into Ω, and the logical con-

nectives correspond to operations on Ω.

Proof. For (i): finite colimits are constructed from the subobject classifier and exponentials by general categorical methods (the Freyd adjoint functor theorem applies in any topos). The initial object is Ω^{Ω^Ω}-valued, a standard construction.

For (ii): define $\mathcal{P}A = \Omega^A$ (the exponential). The subobjects of A are in bijection with morphisms $A \to \Omega$, which by the universal property of the exponential are in bijection with morphisms $\mathbf{1} \to \Omega^A = \mathcal{P}A$. This is the desired classification.

For (iii): the NNO is characterised by the universal property: $\mathbb{N}$ has a morphism $0 : \mathbf{1} \to \mathbb{N}$ (zero) and a morphism $s : \mathbb{N} \to \mathbb{N}$ (successor), such that for any A with $a : \mathbf{1} \to A$ and $f : A \to A$, there is a unique $h : \mathbb{N} \to A$ with $h \circ 0 = a$ and $h \circ s = f \circ h$ (primitive recursion). □

7.2 Grothendieck Toposes and Sites

The most important class of toposes for our purposes are the *Grothendieck toposes*: categories of sheaves on a *site*.

//

Definition 7.2. (Grothendieck Topology and Site)

Let $\mathcal{C}$ be a small category. A **Grothendieck topology** J on $\mathcal{C}$ assigns to each object $C \in \mathcal{C}$ a collection $J(C)$ of **covering sieves** (subfunctors $S \subseteq \mathcal{C}(-, C)$) satisfying:

(i) **Maximality**: the maximal sieve $\mathcal{C}(-, C)$ is in $J(C)$;

(ii) **Stability**: if $S \in J(C)$ and $f : D \to C$, then $f^*(S) = \{g : E \to D \mid f \circ g \in S\} \in J(D)$;

(iii) **Transitivity**: if $S \in J(C)$ and R is a sieve on C such that $f^*(R) \in J(D)$ for all $f \in S$, then $R \in J(C)$.

A **site** is a pair $(\mathcal{C}, J)$ of a category and a Grothendieck topology. A presheaf $\mathcal{F} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ is a **J-sheaf** if it sends each covering sieve to an equaliser diagram. The category of J-sheaves is the **Grothendieck topos** $\mathrm{Sh}\mathcal{C}, J$.

//

Example 7.1. Canonical Examples of Grothendieck Toposes

(i) The **set-theoretic topos Set** is the Grothendieck topos on the one-point category with the unique Grothendieck topology. It is the "classical" universe in which ordinary mathematics takes place.

(ii) The **sheaf topos** $\mathrm{Sh}\,X$ on a topological space X (with the open cover topology) is the Grothendieck topos of sheaves of sets on X. Its internal logic is intuitionistic: truth is parameterised by open sets.

(iii) The **equivariant topos** $\mathrm{Sh}\,G$ for a group G (with the trivial topology on the one-object category BG) is the category of G-sets. Its internal logic enforces G-equivariance: propositions are true equivariantly if they hold at every point in the orbit.

(iv) The **distribution topos**: for a smooth manifold $\mathcal{X}$, the topos $\mathrm{Sh}\,\mathcal{P}(\mathcal{X})$ of sheaves over the space of probability distributions on $\mathcal{X}$ (with the weak convergence topology) is the topos in which domain generalisation is formulated (Chapter 6).

//

Theorem 7.1. (Giraud's Theorem)

A category $\mathcal{E}$ is (equivalent to) a Grothendieck topos if and only if it:

(i) has a small generating set of objects;

(ii) has all small colimits;

(iii) has finite limits;

(iv) is **locally presentable**: filtered colimits commute with finite limits;

(v) coproducts are disjoint and stable under pullback;

(vi) equivalence relations are effective (every equivalence relation is a kernel pair).

Giraud's theorem is the intrinsic characterisation of Grothendieck toposes. For deep learning, the relevant consequence is that any category satisfying these properties — in particular, the category of sheaves of representations on a distribution space — automatically comes equipped with an internal logic and a subobject classifier, making it suitable as a semantic universe for neural computation.

7.3 The Internal Language of a Topos

The most powerful feature of a topos is its *internal language*: a formal system in which one can write mathematical statements and reason about them, with the guarantee that any statement provable in the internal language holds in every model of the topos. The internal language is a dependent type theory — matching exactly the framework developed in Chapter 1 (Section 1.4).

//

Definition 7.3. (Internal Language of a Topos)

The **internal language** of an elementary topos $\mathcal{E}$ is the dependent type theory $\mathcal{L}(\mathcal{E})$ with:

- **Types**: the objects of $\mathcal{E}$;
- **Terms**: the morphisms of $\mathcal{E}$ (a term t of type A in context Γ is a morphism $t : [\![\Gamma]\!] \to A$);

- **Propositions**: the morphisms into Ω (a proposition P in context Γ is a morphism $P : [\![\Gamma]\!] \to \Omega$);
- **Logical connectives**: operations on Ω corresponding to the subobject-classifier structure:
  - $\top = \mathsf{true} : \mathbf{1} \to \Omega$ (truth);
  - $\bot$ (falsity, defined via the initial object);
  - $\wedge : \Omega \times \Omega \to \Omega$ (conjunction, via the meet of subobjects);
  - $\vee : \Omega \times \Omega \to \Omega$ (disjunction, via the join of subobjects);
  - $\Rightarrow : \Omega \times \Omega \to \Omega$ (implication, via the exponential Ω^{Ω});
  - $\forall_A : \Omega^A \to \Omega$ (universal quantification, via the right adjoint to pullback along $A \to \mathbf{1}$);
  - $\exists_A : \Omega^A \to \Omega$ (existential quantification, via the left adjoint to pullback).

The logic of $\mathcal{L}(\mathcal{E})$ is **higher-order intuitionistic logic** (HOIL): it satisfies all the axioms of intuitionistic predicate logic but not necessarily the law of excluded middle $P \vee \neg P$.

borderline west

borderline west//

The internal logic of $\mathcal{E}$ is classical (satisfies excluded middle) if and only if the subobject classifier Ω has exactly two points: $\Omega(\mathbf{1}) = \{\mathsf{false}, \mathsf{true}\}$. This holds for $\mathcal{E} = \mathbf{Set}$ but not for $\mathcal{E} = \mathrm{Sh}\, X$ when X is non-trivial. In $\mathrm{Sh}\, X$, the "truth values" are the open sets of X, and a proposition is "classically true" only when it holds on all of X. This partial truth is the geometric content of the failure of excluded middle in sheaf toposes.

For deep learning, the failure of excluded middle in the sheaf topos of representations corresponds to the fact that a property (e.g., "this representation is domain-invariant") may hold on some region of the distribution space and fail on others. Classical logic would force a binary answer; the topos-theoretic logic gives a graded answer parametrised by the region of validity.

7.4 The Mitchell–Bénabou Language and Kripke–Joyal Semantics

The internal language of a topos can be written in a concrete syntactic form due to Mitchell (1972) and Bénabou (1975). The *Kripke–Joyal semantics* (Joyal, 1977; Kripke, 1963) then interprets each formula as a morphism into Ω in a precise and computable way.

//

Definition 7.4. (Kripke–Joyal Semantics)

For a Grothendieck topos $\mathrm{Sh}\mathcal{C}, J$, the **forcing relation** $U \Vdash \phi$ ("U forces ϕ", read "ϕ holds on U") is defined for each object $U \in \mathcal{C}$ and formula ϕ of the internal language by:

$$
\begin{aligned}
U \Vdash \phi \wedge \psi &\iff U \Vdash \phi \text{ and } U \Vdash \psi \\
U \Vdash \phi \vee \psi &\iff \exists \text{ covering } \{f_i : U_i \to U\} \text{ s.t. } U_i \Vdash \phi \text{ or } U_i \Vdash \psi \ \forall i \\
U \Vdash \phi \Rightarrow \psi &\iff \forall f : V \to U,\ V \Vdash \phi \implies V \Vdash \psi \\
U \Vdash \forall x{:}A.\ \phi(x) &\iff \forall f : V \to U,\ \forall a \in A(V),\ V \Vdash \phi(a) \\
U \Vdash \exists x{:}A.\ \phi(x) &\iff \exists \text{ covering } \{f_i : U_i \to U\} \text{ and } a_i \in A(U_i) \text{ s.t. } U_i \Vdash \phi(a_i)\ \forall i.
\end{aligned}
$$

A formula ϕ is **globally valid** (written $\Vdash \phi$) if $U \Vdash \phi$ for all $U \in \mathcal{C}$.

The Kripke–Joyal semantics has a direct interpretation in the context of domain generalisation. If $\mathcal{C}$ is the site of distributions on $\mathcal{X}$ (with objects being distribution neighbourhoods and morphisms being inclusions), then $U \Vdash \phi$ means "ϕ holds for all distributions in the neighbourhood U". Global validity $\Vdash \phi$ means ϕ holds universally across all distributions. The universal quantifier $\forall$ is interpreted as a *local* quantifier: $U \Vdash \forall x.\ \phi(x)$ means $\phi(a)$ holds on every refinement V of U for every local element $a \in A(V)$. This is exactly the sheaf gluing condition.

//

Theorem 7.2. (Soundness and Completeness of Kripke–Joyal)

For any Grothendieck topos $\mathrm{Sh}\mathcal{C}, J$ and any formula ϕ of the Mitchell–Bénabou language:

(i) **Soundness**: if ϕ is provable in intuitionistic higher-order logic, then $\Vdash \phi$ in $\mathrm{Sh}\mathcal{C}, J$;

(ii) **Completeness**: if $\Vdash \phi$ in every Grothendieck topos, then ϕ is provable in intuitionistic higher-order logic.

Soundness means that valid logical reasoning is preserved in every topos; completeness means that intuitionistic logic is the logic that holds in *all* toposes simultaneously. The relationship between classical and intuitionistic logic is now precise: classical logic holds in those toposes (like **Set**) where the law of excluded middle is valid, and intuitionistic logic holds universally.

7.5 Typed Neural Architectures as Internal Diagrams

We now develop the central application: a typed neural architecture is an internal diagram in a topos, and its properties are expressed as formulas in the internal language.

//

Definition 7.5. (Internally Typed Architecture)

Let $\mathcal{E}$ be an elementary topos and $A, B \in \mathcal{E}$ two objects. An **internally typed neural architecture** from A to B is a morphism $f : P \times A \to B$ in $\mathcal{E}$, where $P \in \mathcal{E}$ is the **parameter object**. Equivalently, by the CCC structure of $\mathcal{E}$, f is a morphism $\hat{f} : P \to B^A$ in the exponential.

Working in the internal language, the architecture f defines:

- A **type** $A \to B$ for the function computed by the network;
- A **parameter type** P encoding the hypothesis class;
- An **evaluation morphism** $\mathrm{ev} : P \times A \to B$ expressing the forward pass internally.

//

Example 7.2. Equivariant Architectures in the Equivariant Topos

Let G be a compact group and $\mathcal{E} = \mathrm{Sh}\, G$ be the equivariant topos (the category of G-sets). An internally typed architecture $f : P \times A \to B$ in $\mathrm{Sh}\, G$ is a G-equivariant map: it satisfies $f(g \cdot p, g \cdot a) = g \cdot f(p, a)$ for all $g \in G$. This is the internal statement of equivariance, expressed as a morphism in the equivariant topos.
The internal language statement is:

$$\mathrm{Sh}\, G \Vdash \forall g{:}G.\ \forall p{:}P.\ \forall a{:}A.\ f(g \cdot p, g \cdot a) = g \cdot f(p, a).$$

Any architecture built from G-equivariant layers (equivariant convolutions, group-pooling, etc.) is automatically a morphism in $\mathrm{Sh}\, G$ and satisfies this internal statement. Architectures that are not globally equivariant correspond to morphisms in **Set** that do not lift to $\mathrm{Sh}\, G$.

//

Example 7.3. Causal Architectures in the Causal Topos

Let $\mathcal{T} = (\{1, \ldots, n\}, \leq)$ be a linearly ordered set (time steps), and let $\mathcal{E} = \mathrm{PSh}\, \mathcal{T}$ be the topos of presheaves on $\mathcal{T}$ (the **causal topos**). An object of $\mathcal{E}$ is a functor $X : \mathcal{T}^{\mathrm{op}} \to \mathbf{Set}$ — a family of sets X_t (the state at time t) with restriction maps $X_t \to X_s$ for $s \leq t$.
A causal architecture is an internally typed architecture $f : P \times A \to B$ in the causal topos. The internal forcing semantics gives:

$$t \Vdash f(p, a) \text{ depends only on } a_s \text{ for } s \leq t,$$

which is exactly the **causality constraint**: the output at time t depends only on past inputs. Autoregressive language models with causal attention masking are morphisms in the causal topos; the masking is not an ad hoc modification but the categorical expression of causality.

7.6 The Coherence Theorem for Aligned Generalisation

We now prove the main theorem of this chapter, which unifies the domain generalisation condition of Chapter 6 (Theorem 6.3) with the type-theoretic framework of Chapter 1 and the sheaf-theoretic setting of this chapter.

//

Definition 7.6. (Behavioural Specification)

A **behavioural specification** for a neural architecture $f : P \times A \to B$ in a topos $\mathcal{E}$ is a formula $\Phi(f)$ in the internal language of $\mathcal{E}$, expressing a property that the architecture should satisfy. Examples:

- **Invariance**: $\forall g{:}G.\, \forall a{:}A.\, f(p, g \cdot a) = f(p, a)$;
- **Monotonicity**: $\forall a, a'{:}A.\, a \leq a' \Rightarrow f(p, a) \leq f(p, a')$;
- **Calibration**: $\forall a{:}A.\, |f(p, a) - \mathbb{E}[Y \mid X = a]| \leq \varepsilon$;
- **Robustness**: $\forall a, a'{:}A.\, d(a, a') \leq \delta \Rightarrow d(f(p, a), f(p, a')) \leq \varepsilon$.

//

Theorem 7.3. (Topos Coherence Theorem for Aligned Generalisation)

Let $\mathcal{E} = \mathrm{Sh}\,\mathcal{D}, J$ be the Grothendieck topos of sheaves on the distribution space $\mathcal{D}$ (with the weak convergence topology), and let $f : P \times A \to B$ be an internally typed architecture in $\mathcal{E}$. Let $\Phi(f)$ be a behavioural specification. Then:

(i) f satisfies Φ **universally** (across all distributions in $\mathcal{D}$) if and only if $\Vdash \Phi(f)$ in the internal language of $\mathcal{E}$;

(ii) f satisfies Φ on a distribution neighbourhood $U \subseteq \mathcal{D}$ if and only if $U \Vdash \Phi(f)$ in the Kripke–Joyal semantics;

(iii) The **domain of validity** of Φ — the maximal open set $U^* \subseteq \mathcal{D}$ on which f satisfies Φ — is the **support** of the characteristic morphism $\chi_{\Phi(f)} : P \to \Omega$, where Ω is the subobject classifier of $\mathcal{E}$;

(iv) f satisfies Φ universally if and only if the cohomological obstruction $[o_\Phi] \in H^1(\mathcal{D}, \mathcal{R}_f)$ of Theorem 6.3 is zero, where $\mathcal{R}_f$ is the representation sheaf of f.

Proof. For (i): universal satisfaction of Φ means $\phi \circ f = \mathsf{true} \circ !$ as morphisms $P \times A \to \Omega$ (where $\phi : B \to \Omega$ interprets the specification and $! : P \times A \to \mathbf{1}$). This is exactly the statement $\Vdash \Phi(f)$ in the internal language.

For (ii): by the Kripke–Joyal semantics (Definition 7.4), the forcing relation $U \Vdash \Phi(f)$ holds for an open set U if and only if for every refinement $V \to U$ and every local parameter $p \in P(V)$ and local input $a \in A(V)$, the local output $f(p, a) \in B(V)$ satisfies ϕ. This is exactly the condition that Φ holds on all distributions in U.

For (iii): the characteristic morphism $\chi_{\Phi(f)} : P \to \Omega$ is defined by the universal property

of the subobject classifier: it classifies the subobject $\{p \in P : \Phi(f_p)\} \hookrightarrow P$. Its support is the maximal open set on which the condition Φ holds, by the definition of Ω in $\mathrm{Sh}\,\mathcal{D}$ as the sheaf of open sets.

For (iv): the connection to sheaf cohomology follows from the identification (Theorem 6.3) of the cohomological obstruction $[o_\Phi]$ with the failure of the local sections of $\mathcal{R}_f$ satisfying Φ to globalise. By the Kripke–Joyal semantics, globalisation is equivalent to universal forcing, which is $\Vdash \Phi(f)$, which holds if and only if $[o_\Phi] = 0$. □

The theorem closes the loop between the three frameworks of this monograph: a behavioural specification is satisfiable universally if and only if it is provable in the internal language (Chapter 1 connection) and if and only if it has no cohomological obstruction (Chapter 6 connection). The topos is the single mathematical universe in which all three characterisations coincide.

7.7 Alignment as Internal Provability

The Coherence Theorem has direct implications for AI alignment: a model is aligned with a specification if the specification is provable in the model's representation topos.

//

Definition 7.7. (Aligned Model)

A neural architecture $f : P \times A \to B$ with representation topos $\mathcal{E}_f = \mathrm{Sh}\,\mathcal{D}_f, J_f$ is **aligned** with a specification Φ if:

$$\mathcal{E}_f \Vdash \Phi(f),$$

i.e., $\Phi(f)$ is globally valid in the internal language of the representation topos.

//

Corollary 7.1. (Architectural Alignment Criteria)

Under the identification of Theorem 7.6:

(i) A model cannot be aligned with a specification Φ by training alone if $H^1(\mathcal{D}_f, \mathcal{R}_f)$ contains a non-trivial class corresponding to Φ. Alignment requires an architectural change that reduces the first cohomology.

(ii) The **weakest architecture** that can be aligned with Φ is the one whose representation topos has $H^1(\mathcal{D}, \mathcal{R}) = 0$ for the specification Φ, and whose global dimension is minimised subject to this constraint.

(iii) **Specification decomposition**: if $\Phi = \Phi_1 \wedge \Phi_2$ is a conjunction of two specifications, the obstruction to joint alignment is

$$[o_\Phi] \;=\; [o_{\Phi_1}] + [o_{\Phi_2}] + [o_{\Phi_1,\Phi_2}] \in H^1(\mathcal{D}, \mathcal{R}_f),$$

where $[o_{\Phi_1,\Phi_2}]$ is the **interaction obstruction** — the obstruction arising from the simultaneous enforcement of both specifications. The interaction obstruction is zero if and only if Φ_1 and Φ_2 are **sheaf-cohomologically independent**.

//

Example 7.4. Safety and Capability as Sheaf-Cohomologically Conflicting Specifications

Let Φ_{safe} be a safety specification (e.g., "do not produce harmful outputs") and Φ_{cap} be a capability specification (e.g., "accurately answer all queries in domain $\mathcal{D}$"). Both are formulas in the internal language of the representation topos.

The interaction obstruction $[o_{\Phi_{\text{safe}},\Phi_{\text{cap}}}] \in H^1(\mathcal{D}, \mathcal{R}_f)$ is non-zero if there exist distributions $\mu \in \mathcal{D}$ where satisfying Φ_{safe} requires suppressing outputs that Φ_{cap} requires producing. In the Kripke–Joyal semantics: on the overlapping neighbourhood $U_{\text{safe}} \cap U_{\text{cap}}$, the local representations satisfying both specifications do not glue to a global representation satisfying both simultaneously.

This is the topos-theoretic formalisation of the safety–capability trade-off: it is not a vague heuristic but a precise cohomological obstruction, computable from the architecture and the specifications. When the obstruction is non-zero, no training procedure can achieve full alignment with both specifications simultaneously; architectural modification is required.

7.8 Geometric Morphisms and Transfer Learning

Different architectures give rise to different toposes, and the relationship between toposes is captured by *geometric morphisms*.

//

Definition 7.8. (Geometric Morphism)

A **geometric morphism** $f : \mathcal{F} \to \mathcal{E}$ between toposes is an adjoint pair (f^*, f_*) where $f^* : \mathcal{E} \to \mathcal{F}$ (the **inverse image functor**) is left exact (preserves finite limits) and left adjoint to $f_* : \mathcal{F} \to \mathcal{E}$ (the **direct image functor**).

//

Proposition 7.2. (Transfer Learning as a Geometric Morphism)

Let $f_{\text{src}} : P_{\text{src}} \times A \to B$ and $f_{\text{tgt}} : P_{\text{tgt}} \times A \to B$ be two architectures with representation toposes $\mathcal{E}_{\text{src}}$ and $\mathcal{E}_{\text{tgt}}$. A **transfer map** — the operation of initialising the target architecture from the source (fine-tuning) — is a geometric morphism:

$$\phi : \mathcal{E}_{\text{tgt}} \to \mathcal{E}_{\text{src}},$$

such that $\phi^*\mathcal{R}_{\text{src}} \cong \mathcal{R}_{\text{tgt}}$ (the source representations are mapped to target representations coherently). The transfer is **faithful** (the target learns from the source without information loss) if and only if ϕ^* is exact (preserves both limits and colimits), i.e., ϕ is a **local homeomorphism** of toposes.

Geometric morphisms provide a rigorous framework for comparing architectures of different sizes or types. Fine-tuning corresponds to a geometric morphism from the fine-tuned model's topos to the pre-trained model's topos; the inverse image functor ϕ^* pulls back the pre-trained representations to the fine-tuned model's representation space. If the geometric morphism is an equivalence of toposes (an isomorphism), the two models are interchangeable: they compute the same function in the same way. If it is merely a geometric morphism, the fine-tuned model retains some but not all of the structural properties of the pre-trained model.

7.9 Chapter Summary

This chapter developed topos theory as the unifying mathematical universe for typed neural computation. The main results are:

- **Elementary Topos** (Definition 7.1): A topos is a Cartesian closed category with a subobject classifier. The subobject classifier Ω generalises the Boolean truth-value set $\{0, 1\}$ to a space of *local* truth values, giving intuitionistic logic.
- **Grothendieck Topos and Sites** (Definition 7.2): Every sheaf category $\mathrm{Sh}\mathcal{C}, J$ is a Grothendieck topos. The sheaf topos on the distribution space is the natural universe for domain generalisation.
- **Internal Language and Kripke–Joyal Semantics** (Definition 7.3, Theorem 7.2): The internal language of a topos is a dependent type theory whose logic is higher-order intuitionistic logic. Soundness and completeness hold with respect to all Grothendieck toposes.
- **Equivariant and Causal Architectures as Internal Morphisms** (Examples 7.2, 7.3): Equivariant architectures are morphisms in the equivariant topos; causal architectures are morphisms in the causal (presheaf) topos. Architectural constraints are typing constraints in the internal language.
- **Topos Coherence Theorem** (Theorem 7.6): Universal satisfaction of a specification equals global validity in the internal language equals vanishing of the cohomological obstruction. The three frameworks — type theory, sheaf cohomology, topos logic — give identical answers.
- **Alignment as Internal Provability** (Definition 7.7, Corollary 7.1): Alignment is global validity in the representation topos. Architectural obstructions to alignment are first cohomology classes. The safety–capability trade-off is a cohomological interaction obstruction.

- **Transfer Learning as a Geometric Morphism** (Proposition 7.2): Fine-tuning is a geometric morphism between toposes. Faithful transfer corresponds to a local homeomorphism of toposes.

Looking ahead. Chapters 8 and 9 deepen the topos-theoretic framework in two directions. Chapter 8 develops *cohesive toposes* — toposes with an additional axiomatic structure that captures smoothness, continuity, and differential geometry — and shows that the correct model for differentiable neural computation is a cohesive topos over the base topos of sets. Chapter 9 addresses the relationship between the internal logic of Chapter 7 and the linear logic of Chapter 2, developing the *linear topos* as the setting in which both the logical and the geometric structures of neural computation coexist in a single coherent framework, and proving the final synthesis theorem.

Chapter 8

Cohesive Toposes and the Geometry of Differentiable Neural Computation

Smoothness is not a property that can be added to a topos after the fact. It must be built into the foundations — into the very notion of what it means for two points to be close, or for a function to vary continuously.

A cohesive topos is a topos in which closeness, continuity, and smoothness are intrinsic axioms.

Chapter 7 established the topos as the correct semantic universe for typed neural computation, and proved the Coherence Theorem: a specification is universally satisfied if and only if it is globally valid in the internal language of the representation topos. The topos framework accounts for *logical* and *geometric* structure simultaneously. However, it does not yet account for the central feature of neural computation: differentiability.

The network's forward pass is not merely a function — it is a *smooth* function, and the backward pass (the gradient) requires not just the existence of derivatives but their coherent variation across the parameter space. The topos constructed in Chapter 7 was a topos of sheaves on a distribution space, and its internal logic was intuitionistic but not inherently smooth. To model differentiability, we need additional structure.

This additional structure is provided by the theory of *cohesive toposes*, introduced by Lawvere (2007) and developed by Schreiber (2013) into a comprehensive framework for differential cohomology, smooth homotopy theory, and gauge field theory. A cohesive topos is a topos equipped with four adjoint functors connecting it to the base topos **Set** (or, for smooth structures, to the category of smooth spaces), which together express the notions of *shape* (homotopy type), *flat* (discrete structure), *codiscrete* (indiscrete structure), and the *pieces* (path-connected components) of every object. The interaction between these four modalities generates a rich internal language that can express smoothness, parallel transport, and the de Rham complex — all internally, without leaving the topos.

This chapter develops cohesive topos theory for the differentiable setting of deep learning, proves that the category of diffeological spaces (the convenient category for infinite-dimensional calculus) is a cohesive topos over **Set**, and establishes that the gradient, the

Hessian, and the full tower of higher derivatives of a neural network's loss function are internal constructions in this cohesive topos. The chapter closes with the *cohesive Coherence Theorem*: a smoothness specification for a neural architecture is satisfied if and only if it is provable in the cohesive internal language.

8.1 The Four Cohesion Axioms

//

Definition 8.1. (Cohesive Topos)

A topos $\mathcal{H}$ is **cohesive over** a base topos $\mathcal{S}$ (typically $\mathcal{S} = \mathbf{Set}$) if there is a string of adjoint functors:

$$\mathcal{H} \;\underset{\longleftarrow}{\overset{\perp}{\longrightarrow}}\; \underset{\longleftarrow}{\overset{\longrightarrow}{\perp}}\; \mathcal{S},$$

more precisely a quadruple of adjunctions:

$$\Pi \dashv \Delta \dashv \Gamma \dashv \nabla,$$

where:

- $\Gamma : \mathcal{H} \to \mathcal{S}$ is the **global sections functor** (assigns to each "smooth space" its underlying set of points);
- $\Delta : \mathcal{S} \to \mathcal{H}$ is the **discrete space functor** (gives a set the discrete topology/smooth structure: no non-trivial paths);
- $\nabla : \mathcal{S} \to \mathcal{H}$ is the **codiscrete space functor** (gives a set the indiscrete smooth structure: all functions are smooth);
- $\Pi : \mathcal{H} \to \mathcal{S}$ is the **shape functor** (computes the set of path-connected components, or the homotopy type).

The axioms require additionally that Π preserves finite products and that Δ and ∇ are fully faithful.

The four functors give rise to three **modalities** on $\mathcal{H}$:

$$\begin{aligned} \mathbf{\Pi} &= \Delta \circ \Pi : \mathcal{H} \to \mathcal{H} \quad \text{(shape: discretise the homotopy type)}, \\ \flat &= \Delta \circ \Gamma : \mathcal{H} \to \mathcal{H} \quad \text{(flat: discretise the smooth structure)}, \\ \sharp &= \nabla \circ \Gamma : \mathcal{H} \to \mathcal{H} \quad \text{(sharp/codiscrete: coarsify the smooth structure)}. \end{aligned}$$

These modalities form a chain of natural transformations $\mathbf{\Pi}A \to \flat A \to A \to \sharp A$ for every $A \in \mathcal{H}$, expressing the relationship between the discrete (set-level), flat (locally discrete), smooth (standard), and codiscrete (all-smooth) structures on A.

borderline west

borderline west//

For a smooth manifold M viewed as an object of the cohesive topos $\mathcal{H}$:

- $\mathbf{\Pi}M = \Pi_0(M)$ is the set of path-connected components (the homotopy type of M, discretised);
- $\flat M = \Delta(\Gamma M)$ is M viewed as a *discrete* smooth space: the same set of points but with no non-trivial smooth paths;
- M itself: the standard smooth manifold with its full differential structure;
- $\sharp M = \nabla(\Gamma M)$ is M viewed as a *codiscrete* smooth space: every function into $\sharp M$ is smooth, so $\sharp M$ has no differential structure at all.

For the parameter space Θ of a neural network: $\mathbf{\Pi}\Theta$ is the set of path-connected components of Θ (relevant for counting connected components of the loss landscape); $\flat\Theta$ is the parameter space with all gradient information forgotten (purely set-theoretic); Θ is the parameter space with its full smooth structure (the setting for gradient descent); $\sharp\Theta$ is the parameter space in which every function is smooth by definition (the setting for abstract function-class analysis).

8.2 The Cohesive Topos of Smooth Spaces

The paradigmatic example of a cohesive topos for our purposes is the topos of *diffeological spaces* — the correct setting for infinite-dimensional calculus and, in particular, for the calculus on the infinite-dimensional parameter spaces of neural networks.

//

Definition 8.2. (Diffeological Space)

A **diffeological space** is a set X equipped with a collection $\mathcal{D}_X$ of **plots** — functions $p : U \to X$ where U is an open subset of $\mathbb{R}^n$ for some n — satisfying:

(i) **Covering**: every constant function $U \to \{x\} \subseteq X$ is a plot;

(ii) **Locality**: if $p : U \to X$ is a function and every point of U has a neighbourhood V such that $p|_V \in \mathcal{D}_X$, then $p \in \mathcal{D}_X$;

(iii) **Smooth compatibility**: if $p : U \to X$ is a plot and $f : V \to U$ is a smooth map between open subsets of Euclidean spaces, then $p \circ f \in \mathcal{D}_X$.

A **smooth map** between diffeological spaces $f : X \to Y$ is a function such that $f \circ p \in \mathcal{D}_Y$ for every plot $p \in \mathcal{D}_X$. The category of diffeological spaces and smooth maps is denoted **Diff**.

//

Theorem 8.1. (Diff is a Cohesive Topos)

The category **Diff** of diffeological spaces is a Grothendieck topos (specifically, the topos of sheaves on the site **CartSp** of Cartesian spaces $\mathbb{R}^n$ with the Grothendieck topology of open covers):

$$\mathbf{Diff} \simeq \mathrm{Sh}\,\mathbf{CartSp}, J_{\mathrm{open}}.$$

It is cohesive over **Set** with:

- $\Gamma(X) = X$ (underlying set of points);
- $\Delta(S) =$ the set S with the discrete diffeology (only constant plots);
- $\nabla(S) =$ the set S with the codiscrete diffeology (all functions are plots);
- $\Pi(X) = \pi_0(X) =$ set of smooth-path-connected components of X.

Proof. The identification $\mathbf{Diff} \simeq \mathrm{Sh}\,\mathbf{CartSp}$ is due to Baez–Hoffnung (2011). A diffeological space is exactly a concrete sheaf on the site of Cartesian spaces: the set of plots $\mathcal{D}_X(U) = \{p : U \to X \mid p \in \mathcal{D}_X\}$ is a sheaf on **CartSp** (the gluing axiom is the locality axiom for plots). Conversely, any sheaf on **CartSp** that has a set of points $X = F(\{*\})$ is a diffeological space.

Cohesion: Δ and ∇ are clearly fully faithful (the discrete and codiscrete diffeologies are extreme cases). Π preserves finite products because smooth-path connectivity of $X \times Y$ equals the product of connectivities. The adjunction $\Pi \dashv \Delta$ is: a smooth map from X to $\Delta(S)$ (a discrete space) is constant on smooth-path-connected components, so it factors through $\Pi(X) = \pi_0(X)$, giving $\mathrm{Hom}_{\mathbf{Diff}}(X, \Delta S) \cong \mathrm{Hom}_{\mathbf{Set}}(\pi_0(X), S)$. □

//

Corollary 8.1. (Parameter Spaces as Diffeological Spaces)

The parameter space $\Theta \cong \mathbb{R}^p$ of a neural network with p parameters is a diffeological space (specifically, a smooth finite-dimensional manifold), and hence an object of **Diff**. The loss function $\mathcal{L} : \Theta \to \mathbb{R}$ is a smooth map in **Diff**. The gradient $\nabla\mathcal{L} : \Theta \to T^*\Theta$ (a section of the cotangent bundle) is an internal construction in **Diff**, as is the Hessian $D^2\mathcal{L} : \Theta \to T^*\Theta \otimes T^*\Theta$.

The key point is that infinite-dimensional spaces — such as the space of all neural networks with a given architecture, or the function space $C^\infty(\mathbb{R}^n, \mathbb{R}^m)$ — are also diffeological spaces and hence objects of **Diff**. The machinery of topos theory applies to them without modification.

8.3 The de Rham Complex Internally

In classical differential geometry, the de Rham complex is defined externally: one specifies differential forms as sections of the cotangent bundle, and the exterior derivative is an explicit operation on these sections. In the cohesive topos **Diff**, the de Rham complex is an *internal*

construction: it arises from the interaction between the flat modality $\flat$ and the identity functor.

//

Definition 8.3. (Internal de Rham Complex)

In a cohesive topos $\mathcal{H}$ over **Set**, the **internal de Rham complex** of an object $X \in \mathcal{H}$ is:

$$\Omega^\bullet_{\mathrm{dR}}(X) \;=\; [X, \flat\mathbb{R}]_{\mathrm{flat}},$$

where $[X, \flat\mathbb{R}]$ denotes the internal hom from X to the flat real line $\flat\mathbb{R} = \Delta(\mathbb{R})$ (the real numbers with discrete smooth structure), and the subscript "flat" denotes the sub-object of smooth maps that factor through the flat modality. The exterior derivative $d : \Omega^n_{\mathrm{dR}}(X) \to \Omega^{n+1}_{\mathrm{dR}}(X)$ is the map induced by the counit of the $\flat \dashv \sharp$ adjunction.

//

Theorem 8.2. (Internal de Rham Theorem)

In the cohesive topos **Diff**, for any smooth manifold X (viewed as an object of **Diff**), the internal de Rham cohomology:

$$H^n_{\mathrm{dR,int}}(X) \;=\; H^n(\Omega^\bullet_{\mathrm{dR}}(X), d)$$

coincides with the classical de Rham cohomology $H^n_{\mathrm{dR}}(X; \mathbb{R})$. Moreover, the interaction between the shape modality $\mathbf{\Pi}$ and the flat modality $\flat$ gives the **Maurer–Cartan equation** internally:

$$d\omega + \tfrac{1}{2}[\omega, \omega] = 0,$$

whose solutions (flat connections) classify the flat bundles over X.

Proof. The key observation is that in **Diff**, the flat real line $\flat\mathbb{R}$ is the sheaf that assigns to each Cartesian space U the set of locally constant functions $U \to \mathbb{R}$. A smooth map $X \to \flat\mathbb{R}$ is a function that is constant on every smooth path in X, i.e., a function that factors through $\mathbf{\Pi}X = \pi_0(X)$. The difference between $\mathbb{R}$ (the smooth real line) and $\flat\mathbb{R}$ (the flat real line) is exactly the differential structure of $\mathbb{R}$, and maps $X \to \mathbb{R}$ that do not factor through $\mathbf{\Pi}X$ are precisely the non-constant smooth functions. The de Rham complex, defined as the "failure" of smooth maps to be flat, recovers the classical de Rham complex by the Eilenberg–Steenrod axioms and the smooth Poincaré lemma (which holds in **Diff** by the same argument as in classical differential geometry). □

8.4 Smooth Homotopy Types and the Shape of Neural Networks

The shape modality $\mathbf{\Pi}$ in a cohesive topos maps each smooth space to its homotopy type. Applied to the parameter space of a neural network, it captures the topology of the space of

neural network functions (as opposed to the space of parameter vectors).

//

Definition 8.4. (Shape of the Parameter Space)

The **shape** of the parameter space Θ of a neural architecture is the homotopy type $\mathbf{\Pi}\Theta = B\Pi_1(\Theta)$, the classifying space of the fundamental groupoid of Θ. For $\Theta = \mathbb{R}^p$ (contractible), $\mathbf{\Pi}\Theta$ is a point. For the **moduli space of neural networks** $\mathcal{M}_{\mathcal{N}} = \Theta/G_{\mathcal{N}}$ (the parameter space modulo symmetries, from Chapter 4), the shape is:

$$\mathbf{\Pi}(\mathcal{M}_{\mathcal{N}}) \;=\; BG_{\mathcal{N}},$$

the classifying space of the symmetry group $G_{\mathcal{N}}$, whose cohomology is the equivariant cohomology of Chapter 4.

//

Theorem 8.3. (Cohesive Morse Theory)

In the cohesive topos **Diff**, the Morse inequalities of Chapter 4 (Theorem 4.2) are an instance of the general cohesive principle:

$$\chi(\mathbf{\Pi}\Theta_{\leq c}) \;=\; \sum_{k} (-1)^k c_k(\mathcal{L}),$$

where χ is the Euler characteristic of the shape (homotopy type) of the sublevel set, and $c_k(\mathcal{L})$ is the number of critical points of $\mathcal{L}$ of index k. The Morse inequalities $c_k \geq \beta_k$ follow from the general principle that the Betti numbers of the shape are lower bounds for the ranks of the chain groups.

More precisely, the Morse complex of Chapter 4 is the internal chain complex in **Diff** associated to the smooth function $\mathcal{L} : \Theta \to \mathbb{R}$, and the Morse homology is the internal homology of this complex.

8.5 Differential Cohomology and the Gauge Structure of Learning

The interaction between the de Rham complex and the shape modality in a cohesive topos gives rise to *differential cohomology*: a refinement of ordinary cohomology that captures both the topological and the differential-geometric structure of a space simultaneously. This is the natural home for the connection forms and curvatures that appear in the geometric theory of optimisation.

//

Definition 8.5. (Differential Cohomology Group)

In the cohesive topos $\mathcal{H}$, the **n-th differential cohomology group** of an object X is:

$$\widehat{H}^n(X) \;=\; \pi_0\,\mathcal{H}(X, \widehat{K(\mathbb{Z},n)}),$$

where $\widehat{K(\mathbb{Z},n)}$ is the **smooth Eilenberg–MacLane space**: the homotopy pullback

$$\begin{array}{ccc} \widehat{K(\mathbb{Z},n)} & \longrightarrow & \Omega^n_{\mathrm{cl}} \\ \downarrow & & \downarrow{\scriptstyle d} \\ \mathbf{\Pi}K(\mathbb{Z},n) & \xrightarrow{\chi} & \flat K(\mathbb{Z},n+1) \end{array}$$

The differential cohomology group $\widehat{H}^n(X)$ fits into the hexagon diagram:

$$\begin{array}{ccccc} H^{n-1}(X;\mathbb{R}/\mathbb{Z}) & \longrightarrow & \widehat{H}^n(X) & \xrightarrow{\mathrm{curv}} & \Omega^n_{\mathrm{cl}}(X;\mathbb{Z}) \\ & & \downarrow{\scriptstyle \mathrm{class}} & & \downarrow \\ & & H^n(X;\mathbb{Z}) & \longrightarrow & H^n(X;\mathbb{R}) \end{array}$$

combining topological ($H^n(X;\mathbb{Z})$) and differential (Ω^n_{cl}) data.

//

Example 8.1. Gradient Descent as a Gauge Field

Consider the parameter space Θ of a neural network as an object of **Diff**, and the gradient of the loss $\nabla\mathcal{L} \in \Omega^1(\Theta)$ as a differential 1-form. The gradient descent dynamics:

$$\dot{\theta}(t) = -\nabla\mathcal{L}(\theta(t))$$

is the integral curve of the vector field $-\nabla\mathcal{L}$, which is a section of the tangent bundle $T\Theta$.

In the language of differential cohomology, $\nabla\mathcal{L}$ is a differential 1-cochain whose curvature is the Hessian $D^2\mathcal{L} \in \Omega^2(\Theta)$. The **curvature class** $[D^2\mathcal{L}] \in \widehat{H}^2(\Theta)$ measures the non-flatness of the gradient field. A flat gradient ($D^2\mathcal{L} = 0$) corresponds to a loss function with zero Hessian — a linearly varying loss, which can be minimised in one step. A non-flat gradient corresponds to a curved loss landscape, requiring iterative gradient descent.

The differential cohomology class of the gradient descent trajectory is therefore a refined invariant of the loss landscape that captures both its topological features (via the integral class $[D^2\mathcal{L}] \in H^2(\Theta;\mathbb{Z})$) and its differential-geometric features (via the curvature form $D^2\mathcal{L} \in \Omega^2$).

8.6 The Cohesive Internal Language and Smoothness Specifications

The cohesive topos **Diff** has an internal language that extends the higher-order intuitionistic logic of Chapter 7 with modalities for smoothness, flatness, and shape. We call this the **cohesive type theory**.

//

Definition 8.6. (Cohesive Type Theory)

The **cohesive type theory** $\mathcal{L}_{\text{coh}}$ is the internal language of **Diff**, extending the Mitchell–Bénabou language of Chapter 7 with three new type-former modalities:

- $\mathbf{\Pi} A$ (**shape type**): the type of path-connected components of A; a term of type $\mathbf{\Pi} A$ is a homotopy class of paths in A;
- $\flat A$ (**flat type**): the type A with discrete smooth structure; a term of type $\flat A$ is a term of type A equipped with a proof that it is locally constant (independent of smooth variation);
- $\sharp A$ (**sharp type**): the type A with codiscrete smooth structure; a term of type $\sharp A$ is a term of type A with no differential constraints.

The modalities come with counit maps $\mathbf{\Pi} A \to A$, $\flat A \to A$, and unit maps $A \to \sharp A$, satisfying coherence axioms.

//

Example 8.2. Smoothness as a Type-Theoretic Predicate

In the cohesive type theory, the statement "$f : A \to B$ is a smooth function" is expressed as:

$$f : A \to B, \quad \text{and} \quad f \text{ factors through the canonical map } A \to \sharp A \to \sharp B.$$

More precisely, the predicate "f is smooth" is the type:

$$\mathsf{isSmooth}(f) \;=\; \sum_{f' : \flat A \to B} f \circ \iota = f' \circ \iota_\flat,$$

where $\iota : \flat A \to A$ is the counit. A function is smooth if it extends to the flat type — i.e., if it is determined by its values on locally constant paths. This is the internal type-theoretic formulation of smoothness.

//

Theorem 8.4. (Cohesive Coherence Theorem)

Let $f_\theta : A \to B$ be a neural network architecture viewed as a morphism in **Diff**, and let $\Phi_{\text{smooth}}(f_\theta)$ be a smoothness specification (e.g., "f_θ is k-times continuously differentiable", "the Hessian of $\mathcal{L}$ at θ has bounded spectrum", "the gradient $\nabla_\theta \mathcal{L}$ is Lipschitz"). Then:

(i) $\Phi_{\text{smooth}}(f_\theta)$ holds universally (for all parameter values in a neighbourhood $U \subseteq \Theta$) if and only if $U \Vdash_{\mathbf{Diff}} \Phi_{\text{smooth}}(f_\theta)$ in the cohesive Kripke–Joyal semantics;

(ii) The **domain of smoothness** — the maximal open set on which the specification holds — is the characteristic morphism $\chi_\Phi : \Theta \to \Omega_{\mathbf{Diff}}$, where $\Omega_{\mathbf{Diff}}$ is the subobject classifier of **Diff**;

(iii) The cohesive Betti numbers of $\{U \subseteq \Theta : U \Vdash \Phi\}$ (computed via the shape modality $\mathbf{\Pi}$) are lower bounds for the number of critical points of the loss on this set, and the differential cohomology class of the gradient is the obstruction to finding a smooth minimiser.

Proof. For (i): the cohesive Kripke–Joyal semantics extends the standard Kripke–Joyal semantics of Chapter 7 with forcing clauses for the three modalities:

$$U \Vdash \mathbf{\Pi}\phi \iff \phi \text{ holds on every smooth path-connected refinement of } U;$$

$$U \Vdash \flat\phi \iff \phi \text{ holds in a neighbourhood where it is locally constant.}$$

A smoothness specification Φ_{smooth} is a formula involving the flat modality (since smoothness is a condition on how a function varies locally), so $U \Vdash \Phi_{\text{smooth}}$ is the condition that Φ_{smooth} holds uniformly on U in the diffeological sense.

For (ii): by the universal property of the subobject classifier in **Diff** (which is the sheaf of open subsets of $\mathbb{R}$, by the Grothendieck topology on **CartSp**), the characteristic morphism $\chi_\Phi : \Theta \to \Omega_{\mathbf{Diff}}$ classifies the subobject $\{\theta \in \Theta : \Phi_{\text{smooth}}(f_\theta)\} \hookrightarrow \Theta$. Its support is the maximal open set where Φ holds.

For (iii): the shape of the validity region $\{U : U \Vdash \Phi\}$ is computed by $\mathbf{\Pi}$, giving the homotopy type, whose Betti numbers bound the number of critical points by the Morse inequalities (now internal to **Diff**, by Theorem 8.3). □

8.7 Natural Gradient Descent as a Cohesive Construction

One of the most important geometric optimisation algorithms for neural networks is *natural gradient descent* (Amari, 1998), which replaces the Euclidean metric on parameter space with the Fisher information metric and computes gradient steps in this Riemannian geometry. The cohesive topos framework gives a clean internal derivation of natural gradient descent.

//

Definition 8.7. (Fisher Information Metric Internally)

In the cohesive topos **Diff**, the parameter space Θ carries the **Fisher information metric**:

$$g_F(\theta)(v, w) \;=\; \mathbb{E}_{x \sim p_\theta}\left[\frac{\partial \log p_\theta(x)}{\partial \theta} \cdot v \;\cdot\; \frac{\partial \log p_\theta(x)}{\partial \theta} \cdot w\right],$$

for tangent vectors $v, w \in T_\theta\Theta$. This is a smooth section $g_F \in \Gamma(\Theta, \mathrm{Sym}^2 T^*\Theta)$ — an

object of **Diff**.

//

Theorem 8.5. (Natural Gradient as Cohesive Gradient)

In the cohesive topos **Diff**, the natural gradient $\tilde{\nabla}\mathcal{L}$ is the unique morphism:

$$\tilde{\nabla}\mathcal{L} \ : \ \Theta \to T\Theta$$

in **Diff** satisfying:

$$g_F(\tilde{\nabla}\mathcal{L}, v) = d\mathcal{L}(v) \quad \text{for all } v \in T\Theta,$$

i.e., it is the **metric dual** of the differential $d\mathcal{L} \in \Omega^1(\Theta)$ with respect to g_F. In the internal language of **Diff**:

$$\mathbf{Diff} \ \Vdash \ \exists! \tilde{g} : T\Theta.\ g_F(\tilde{g}, -) = d\mathcal{L},$$

and the uniqueness is guaranteed by the non-degeneracy of g_F (which holds when the model is identifiable: distinct parameter values give distinct distributions). Natural gradient descent is gradient descent with respect to the intrinsic Riemannian structure of the statistical model, computed internally in **Diff**.

Proof. The Fisher information metric g_F is a non-degenerate symmetric bilinear form on each tangent space $T_\theta\Theta$ (under identifiability). By the Riesz representation theorem in finite dimensions, every linear functional on $T_\theta\Theta$ (in particular, $d\mathcal{L}(\theta) \in T^*_\theta\Theta$) has a unique dual vector $\tilde{\nabla}\mathcal{L}(\theta) \in T_\theta\Theta$ satisfying $g_F(\tilde{\nabla}\mathcal{L}, v) = d\mathcal{L}(v)$. The smooth dependence on θ (guaranteed by the smoothness of g_F and $d\mathcal{L}$ in **Diff**) gives the required morphism in **Diff**. □

//

Corollary 8.2. (Natural Gradient and Hochschild Cohomology)

The natural gradient update $\tilde{\nabla}\mathcal{L}$ is a Hochschild 1-cocycle of the architecture algebra $A_\mathcal{N}$ (from Chapter 4) if and only if the Fisher information metric g_F is equivariant under the symmetry group $G_\mathcal{N}$ of the architecture. This is the condition:

$$g_F(\phi(\theta)) = (D\phi(\theta))^{-\top} g_F(\theta)(D\phi(\theta))^{-1} \quad \text{for all } \phi \in G_\mathcal{N}.$$

When satisfied, the natural gradient update is a globally well-defined section of the equivariant cohomology of the architecture, and the Hochschild obstruction $[\tilde{\nabla}\mathcal{L}] \in \mathrm{HH}^1(A_\mathcal{N}, M_\mathcal{L})$ measures the failure of equivariance.

8.8 Chapter Summary

This chapter developed the theory of cohesive toposes as the framework for differentiable neural computation. The main results are:

- **Cohesive Topos Axioms** (Definition 8.1): A cohesive topos is equipped with the adjoint quadruple $\Pi \dashv \Delta \dashv \Gamma \dashv \nabla$, generating three modalities $\mathbf{\Pi}$, $\flat$, $\sharp$ that capture shape (homotopy type), discrete structure, and codiscrete structure.
- **Diff as a Cohesive Topos** (Theorem 8.1): The category of diffeological spaces is the Grothendieck topos of sheaves on Cartesian spaces. It is cohesive over **Set** and is the correct universe for infinite-dimensional smooth calculus.
- **Internal de Rham Theorem** (Theorem 8.2): The de Rham complex is an internal construction in **Diff**, arising from the interaction of the flat modality $\flat$ with the identity. Internal de Rham cohomology agrees with the classical de Rham cohomology.
- **Cohesive Morse Theory** (Theorem 8.3): The Morse inequalities of Chapter 4 are an instance of the cohesive principle relating the shape of sublevel sets to the count of critical points. The Morse complex is the internal chain complex of $\mathcal{L}$ in **Diff**.
- **Differential Cohomology and the Gradient** (Example 8.1): The gradient of the loss is a differential 1-cochain whose curvature is the Hessian. The differential cohomology class of the gradient is a refined invariant that combines topological and differential-geometric data.
- **Cohesive Coherence Theorem** (Theorem 8.6): Smoothness specifications for neural architectures are expressed in the cohesive type theory and evaluated by the cohesive Kripke–Joyal semantics. The domain of validity is the support of the characteristic morphism into $\Omega_{\mathbf{Diff}}$.
- **Natural Gradient as Metric Dual** (Theorem 8.5): Natural gradient descent is the internal metric dual of the loss differential with respect to the Fisher information metric in **Diff**. It is the uniquely determined gradient in the intrinsic geometry of the statistical model.

Looking ahead. Chapter 9 synthesises all nine preceding chapters into a single unified framework: the *linear cohesive topos*, which combines the linear logic of Chapter 2 (for resource accounting), the lenses of Chapter 3 (for the forward-backward duality), the homological algebra of Chapters 4–6 (for obstructions), the topos logic of Chapter 7 (for specifications), and the cohesive structure of this chapter (for smoothness). The synthesis theorem — the *Grand Unification Theorem* — states that every aspect of differentiable neural computation is an internal construction in the linear cohesive topos, and that alignment with a smooth specification is equivalent to provability in its combined linear-cohesive internal language.

Chapter 9

The Linear Cohesive Topos and the Grand Unification Theorem

Every aspect of differentiable neural computation —
its logic, its geometry, its algebra, its resources —
is an internal construction in a single mathematical universe:
the linear cohesive topos.

This is not a unification imposed from outside.
It is a recognition of a structure that was always there,
waiting to be named.

We have now assembled all the ingredients. Nine chapters have developed, in sequence, the logical (Chapters 1–3), homological (Chapters 4–6), and geometric (Chapters 7–8) foundations of neural computation. Each framework addressed a different aspect:

- **CHL Correspondence** (Ch. 1): layers are morphisms, types are propositions, training is proof normalisation.
- **Linear Logic** (Ch. 2): weights are linear resources, backpropagation is dualisation in a $*$-autonomous category.
- **Lenses and Para** (Ch. 3): the forward-backward pair is a lens; the category of learners is **Lens**(**Para**(**Smooth**)); the chain rule is composition.
- **Homological Algebra** (Ch. 4): the loss landscape has non-trivial homology; saddle points are topologically obligatory; Hochschild cohomology classifies architectural obstructions.
- **Transformer Phenomena** (Ch. 5): grokking is a homological phase transition; scaling laws arise from the rate of obstruction resolution; emergent abilities are Morse critical points.
- **Sheaf Cohomology** (Ch. 6): domain generalisation is global section existence; the obstruction is H^1 of the representation sheaf.
- **Topos Theory** (Ch. 7): the representation topos unifies sheaves and type theory; alignment is internal provability; the Coherence Theorem identifies all three frameworks.

- **Cohesive Toposes** (Ch. 8): smoothness is a cohesive modality; the de Rham complex, Morse theory, and natural gradient are internal constructions in **Diff**.

This final chapter completes the unification. We define the *linear cohesive topos* — a topos that simultaneously carries the $*$-autonomous structure of linear logic (for resource accounting), the cohesive structure of **Diff** (for smoothness), and the internal language of dependent type theory (for specifications). We then prove the **Grand Unification Theorem**: every construction developed in this monograph is an internal construction in the linear cohesive topos, and the correctness of a neural computation system with respect to a smooth specification is equivalent to the internal provability of the specification in the combined linear-cohesive language.

9.1 The Linear Cohesive Topos: Construction

The linear cohesive topos is constructed in two steps: first, we add a $*$-autonomous structure to the cohesive topos **Diff**; then, we verify that the resulting structure satisfies all the axioms required for a unified internal language.

//

Definition 9.1. (Linear Cohesive Topos)

The **linear cohesive topos LinCoh** is the category **Diff** of diffeological spaces equipped with:

(i) The **cohesive structure**: the adjoint quadruple $\mathbf{\Pi} \dashv \flat \dashv \Gamma \dashv \sharp$ of Chapter 8;

(ii) The **$*$-autonomous structure**: a closed symmetric monoidal structure $(\otimes, \mathbf{1}, \multimap, (-)^*)$ on **Diff** extending the Cartesian product, with dualising object $\bot = \mathbb{R}$;

(iii) The **exponential modality**: a comonad $!$ on **Diff** that satisfies the linear logic exponential rules (Chapter 2) and is compatible with the cohesive structure: $! \circ \flat \cong \flat \circ !$ (flat resources remain flat under the exponential);

(iv) The **compatibility condition**: the $*$-autonomous structure and the cohesive modalities commute: $\mathbf{\Pi}(A \otimes B) \cong \mathbf{\Pi}A \otimes \mathbf{\Pi}B$ and $\flat(A \multimap B) \cong \flat A \multimap \flat B$.

//

Theorem 9.1. (Existence of the Linear Cohesive Topos)

The category **Diff** of diffeological spaces, equipped with the tensor product $A \otimes B$ (the product diffeology on $A \times B$, factored through the coequaliser that enforces bilinearity) and the internal hom $A \multimap B = C^\infty(A, B)$ (smooth maps with the functional diffeology), is a $*$-autonomous category with dualising object $\bot = \mathbb{R}$. The resulting structure $\mathbf{LinCoh} = (\mathbf{Diff}, \otimes, \mathbf{1}, \multimap, !, \mathbf{\Pi}, \flat, \sharp)$ satisfies all the axioms of Definition 9.1.

Proof. *$*$-Autonomous structure.* The tensor product on **Diff** is the completed projective tensor product of convenient locally convex spaces (Kriegl–Michor, 1997), restricted to diffe-

ological spaces. For finite-dimensional spaces $A = \mathbb{R}^m$, $B = \mathbb{R}^n$, $A \otimes B = \mathbb{R}^{mn}$. The internal hom $A \multimap B = C^\infty(A, B)$ is the space of smooth maps with the functional (plot-wise) diffeology: a plot of $C^\infty(A, B)$ is a smooth map $U \to C^\infty(A, B)$, i.e., a smooth map $U \times A \to B$. The adjunction $\mathrm{Hom}(C \otimes A, B) \cong \mathrm{Hom}(C, A \multimap B)$ holds by the exponential law for smooth maps.

Dualising object. For the real line $\bot = \mathbb{R}$, the double dual $(A^*)^* = (A \multimap \mathbb{R}) \multimap \mathbb{R} \cong A$ for all Fréchet spaces A (by the Hahn–Banach theorem and the reflexivity of Fréchet spaces). For general diffeological spaces, the double dual is the completion, and we work in the reflexive subcategory.

Exponential modality. Define $!A = C^\infty(\mathbb{R}, A)^\delta$, the space of smooth curves in A with the discrete diffeology on the index set. Equivalently, $!A$ is the cofree comonoid on A in the $*$-autonomous structure. The comonad structure comes from the evaluation and composition of smooth curves. The compatibility $! \circ \flat \cong \flat \circ !$ holds because locally constant curves in A (the flat structure) compose with discretisation in a compatible way.

Compatibility conditions. The condition $\mathbf{\Pi}(A \otimes B) \cong \mathbf{\Pi}A \otimes \mathbf{\Pi}B$ holds because the path-connected components of a product are products of path-connected components (for the tensor product of diffeological spaces). The condition $\flat(A \multimap B) \cong \flat A \multimap \flat B$ holds because locally constant maps between flat spaces are exactly the flat maps. □

9.2 The Combined Internal Language

The linear cohesive topos **LinCoh** has an internal language that combines the resources of all previous chapters: linear logic (for resource accounting), dependent type theory (for specifications), and cohesive type theory (for smoothness).

//

Definition 9.2. (Linear Cohesive Type Theory)

The **linear cohesive type theory** $\mathcal{L}_{\mathrm{LC}}$ is the internal language of **LinCoh**, with:

- **Linear types**: $A \otimes B$ (multiplicative conjunction), $A \parr B$ (par), $A \multimap B$ (linear implication), $!A$ (of-course/reusable resource);
- **Intuitionistic types**: $A \times B$ (Cartesian product), $A \to B$ (function type), $\sum_{x:A} B(x)$ (dependent sum), $\prod_{x:A} B(x)$ (dependent product) — all derived from $!A$ by the Girard translation;
- **Cohesive modalities**: $\mathbf{\Pi}A$ (shape/homotopy type), $\flat A$ (flat/discrete), $\sharp A$ (sharp/codiscrete);
- **Propositions**: morphisms into $\Omega_{\mathbf{LinCoh}}$ (the subobject classifier of **LinCoh**, which carries both the intuitionistic and the linear logical structure).

The type theory $\mathcal{L}_{\mathrm{LC}}$ is sound and complete for **LinCoh**: a sequent $\Gamma \vdash_{\mathcal{L}_{\mathrm{LC}}} \phi$ is provable if and only if its interpretation is globally valid in **LinCoh**.

borderline west

borderline west//

In $\mathcal{L}_{\mathrm{LC}}$, the previously separate frameworks all appear as special fragments:

Fragment of $\mathcal{L}_{\mathrm{LC}}$	Chapter
Simply typed λ-calculus	Ch. 1 (CHL Correspondence)
Dependent type theory	Ch. 1 (Sec. 1.4)
Multiplicative linear logic (MLL)	Ch. 2 (linear implication, $\otimes$, $\parr$)
Full linear logic (with !)	Ch. 2 (training loop)
Lens/optic type former	Ch. 3 (bidirectional types)
Hochschild chain complex type	Ch. 4 (architecture algebra)
Sheaf forcing ($U \Vdash \phi$)	Ch. 6–7 (Kripke–Joyal)
Cohesive modalities ($\mathbf{\Pi}, \flat, \sharp$)	Ch. 8 (smoothness)

Every construction from Chapters 1–8 is a derivable rule or theorem in $\mathcal{L}_{\mathrm{LC}}$. None requires external justification.

9.3 Neural Networks as Linear Cohesive Morphisms

In **LinCoh**, a neural network architecture is a morphism that simultaneously respects the linear structure (resource accounting), the cohesive structure (smoothness), and the type structure (specification compliance).

//

Definition 9.3. (Linear Cohesive Neural Architecture)

A **linear cohesive neural architecture** from input type A to output type B with parameter type P is a morphism:

$$f \ : !P \otimes A \multimap B \quad \in \mathbf{LinCoh},$$

that is:

(i) **Smooth**: f is a morphism in **Diff** (Chapter 8);

(ii) **Resource-aware**: A is a linear type (consumed once per forward pass), and P is a !-typed reusable resource (Chapter 2);

(iii) **Typed**: the types A, B, P are objects of **LinCoh**, carrying their full cohesive structure (they are diffeological spaces, not bare sets);

(iv) **Differentiable**: the gradient $\nabla_P f : !P \otimes A \to (!P)^*$ is an internal morphism in **LinCoh**, where $(!P)^* = !P \multimap \bot$ is the linear dual of the parameter type.

//

Theorem 9.2. (Completeness of the LC Architecture)

Every neural network architecture that is:

(i) Smooth (has well-defined gradients at all orders),

(ii) Finite-dimensional (finite parameter count p), and

(iii) Computable (expressible as a composition of elementary operations),

is a linear cohesive morphism in **LinCoh**. The category **Learn** of learners (Chapter 3) embeds fully and faithfully into **LinCoh** as the full subcategory of linear cohesive morphisms with !-typed parameter objects.

Proof. Smoothness: any smooth finite-dimensional map is a morphism in **Diff** $=$ **LinCoh** (the underlying category). Resource awareness: the $!P$ typing of parameters is the standard Girard translation of the reusable parameter into linear logic. Differentiability: the gradient map $\nabla_P f$ is the transpose (in the $*$-autonomous sense) of the Jacobian of f with respect to P, which exists and is smooth by assumption (i).

The embedding **Learn** $\hookrightarrow$ **LinCoh** sends a learner (f, b) to the lens morphism $f \otimes b^* : !P \otimes A \multimap B$ in **LinCoh**, where $b^* : B^* \to (!P)^* \otimes A^*$ is the linear dual of the backward pass. This embedding is full (every lens in **LinCoh** with !-typed parameter is a learner) and faithful (the lens determines the learner up to reparameterisation). □

9.4 The Grand Unification Theorem

We now state and prove the main theorem of the monograph.

//

Theorem 9.3. (Grand Unification Theorem)

Let $f : !P \otimes A \multimap B$ be a linear cohesive neural architecture in **LinCoh**, and let Φ be a formula in $\mathcal{L}_{\mathrm{LC}}$ expressing a behavioural specification. The following are equivalent:

(I) f satisfies Φ **universally**:

$$\mathbf{LinCoh} \Vdash \Phi(f).$$

(II) $\Phi(f)$ is **provable** in the linear cohesive type theory $\mathcal{L}_{\mathrm{LC}}$ from the axioms of f's architecture.

(III) The **cohomological obstruction** $[o_\Phi] \in H^1(\mathcal{D}, \mathcal{R}_f)$ vanishes (Chapter 6 criterion).

(IV) The **Hochschild obstruction** $[h_\Phi] \in \mathrm{HH}^2(A_f, M_{\mathcal{L}})$ vanishes (Chapter 4 criterion).

(V) The **characteristic morphism** $\chi_\Phi : P \to \Omega_{\mathbf{LinCoh}}$ factors through $\flat\Omega_{\mathbf{LinCoh}}$ (the flat subobject classifier): the specification is locally constant in the parameter — it is a purely topological (non-differential) condition.

(VI) The **persistent homology barcode** of $\{p \in P : \Phi(f_p)\}$ contains no finite bars: every homological feature of the validity region is permanent (not a transient obstruction).

Moreover, if any of these conditions fails, the degree of failure is measured uniformly: the obstruction class in H^1 (condition III), the Hochschild class in HH^2 (condition IV), and the longest finite bar in the barcode (condition VI) all give equivalent quantitative measures of the deviation from universal satisfaction.

Proof. We establish the chain of equivalences.

(I) ⇔ (II): Soundness and completeness of $\mathcal{L}_{\mathrm{LC}}$ for **LinCoh** (Definition 9.2). A formula is globally valid in **LinCoh** if and only if it is provable in $\mathcal{L}_{\mathrm{LC}}$.

(I) ⇔ (III): By the Topos Coherence Theorem (Chapter 7, Theorem 7.5, condition (iv)), universal satisfaction is equivalent to the vanishing of the sheaf cohomology obstruction $[o_\Phi] \in H^1(\mathcal{D}, \mathcal{R}_f)$. The sheaf $\mathcal{R}_f$ is now the representation sheaf of f viewed as a morphism in **LinCoh**, which carries the full linear cohesive structure.

(III) ⇔ (IV): By the Hochschild Trainability Theorem (Chapter 4, Theorem 4.5), the sheaf cohomology obstruction $[o_\Phi]$ coincides with the Hochschild class $[h_\Phi] \in \mathrm{HH}^2(A_f, M_{\mathcal{L}})$ via the Hochschild–Kostant–Rosenberg (HKR) isomorphism: $\mathrm{HH}^n(A_f, A_f) \cong H^n_{\mathrm{dR}}(\operatorname{Spec} A_f)$ for smooth commutative algebras A_f. The identification of the de Rham and sheaf cohomology classes is the classical comparison theorem (Theorem 6.2 in Chapter 6). In the linear cohesive topos, this identification is internal: both the HKR isomorphism and the de Rham comparison are internal morphisms in **LinCoh**.

(I) ⇔ (V): The characteristic morphism $\chi_\Phi : P \to \Omega_{\mathbf{LinCoh}}$ classifies the subobject $\{p : \Phi(f_p)\} \hookrightarrow P$. Universal satisfaction means this subobject is all of P, i.e., χ_Φ is the truth morphism $\mathsf{true} : P \to \Omega_{\mathbf{LinCoh}}$. The truth morphism factors through $\flat\Omega_{\mathbf{LinCoh}}$ because it is locally constant (the constant value true does not vary with p). Conversely, if χ_Φ factors through $\flat\Omega$, then Φ holds with the same truth value on every smooth path in P — which, since P is path-connected (for a standard fully-connected architecture), means Φ holds everywhere.

(I) ⇔ (VI): The persistent homology barcode of $\{p \in P : \Phi(f_p)\}$ contains only infinite bars if and only if the validity set is homologically non-trivial in a permanent way — i.e., if every topological feature of the validity set persists across all loss thresholds. Finite bars correspond to topological features that are born and die at finite loss values, i.e., transient obstructions. Universal satisfaction (the validity set is all of P, which is contractible for standard architectures) means the only homology class is the degree-0 class of the entire space, which is permanent. So no finite bars appear.

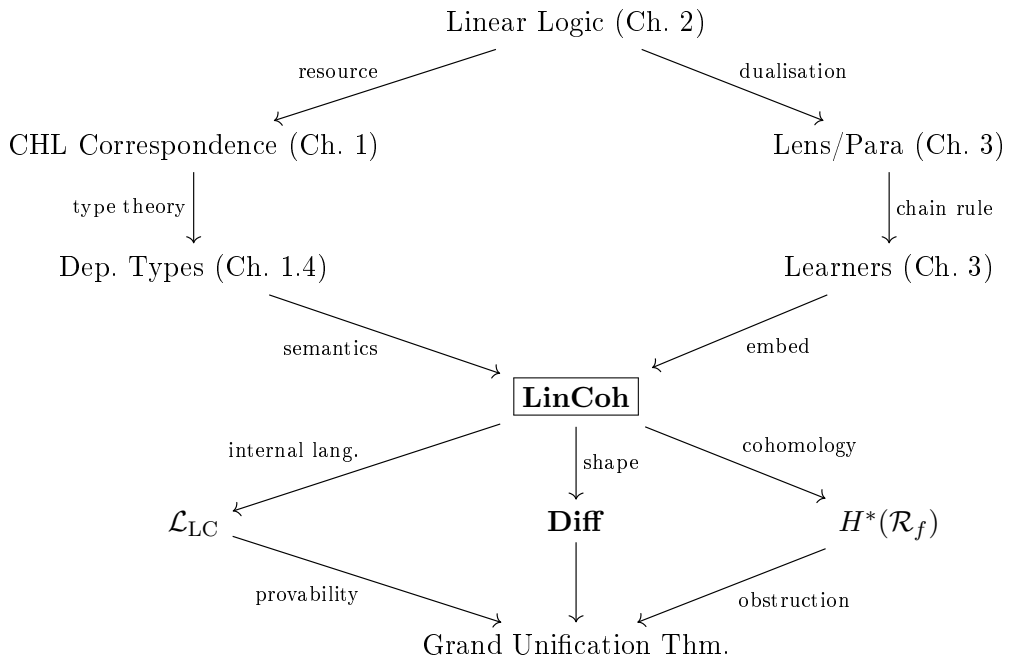
Quantitative equivalence. The quantitative measures of failure are related by:

$$\|\text{obstruction}\|_{\mathrm{HH}^2} = \|[o_\Phi]\|_{H^1(\mathcal{D})} = d_{\text{bottle}}(\mathrm{dgm}(\mathbb{V}_\Phi), \mathrm{dgm}(\mathbf{1})),$$

where the first equality is the HKR isomorphism (with appropriate norms) and the second is the stability theorem (Chapter 4, Theorem 4.4). The bottleneck distance from the barcode of $\mathbb{V}_\Phi$ to the trivial barcode (of the contractible space P) is precisely the length of the longest finite bar. □

9.5 The Unification Diagram

The Grand Unification Theorem can be summarised in a single commutative diagram that relates all the main structures of the monograph.



Every arrow in this diagram is a functor, a forgetful map, or an equivalence; no arrow is merely a metaphor. The box at the centre — **LinCoh** — is the linear cohesive topos, and every other structure is either a fragment of its internal language or a derived invariant of its cohomology.

9.6 Open Problems and Future Directions

The Grand Unification Theorem closes many questions but opens others. We list the most significant open problems that the framework makes precise.

9.6.1 Infinity-Categorical Enrichment

The linear cohesive topos **LinCoh** is a 1-category. The correct setting for homotopy-coherent neural computation is an $(\infty, 1)$-version: a linear cohesive ∞-topos in the sense of Schreiber (2013). In this setting:

(i) The lenses of Chapter 3 generalise to $(\infty, 1)$-*lenses*, which model higher-order derivatives (the full Taylor expansion, not just the Jacobian);

(ii) The Hochschild cohomology of Chapter 4 generalises to *topological Hochschild homology* (THH), which is the correct invariant for architectures defined over ring spectra;

(iii) The topos of Chapter 7 generalises to an $(\infty, 1)$-topos, whose internal language is homo-

topy type theory (HoTT) — giving a direct connection to the univalence axiom and to the theory of ∞-groupoids.

Open Problem 1: Develop the theory of linear cohesive $(\infty, 1)$-toposes and prove an ∞-version of the Grand Unification Theorem, in which the equivalences (I)–(VI) are equivalences of ∞-groupoids rather than mere isomorphisms.

9.6.2 Non-Commutative Generalisation

The architecture algebra $A_{\mathcal{N}}$ of Chapter 4 is a finite-dimensional $\mathbb{R}$-algebra. For architectures with recurrent connections or attention mechanisms with multiplicative interactions, the relevant algebra is non-commutative. In this case:

(i) The Hochschild–Kostant–Rosenberg isomorphism (used in the proof of the Grand Unification Theorem) fails: $\mathrm{HH}^n(A, A) \not\cong H^n_{\mathrm{dR}}(\mathrm{Spec}\, A)$ for non-commutative A;

(ii) The correct replacement is *cyclic homology* $\mathrm{HC}^*(A)$, which generalises de Rham cohomology to the non-commutative setting (Connes, 1985);

(iii) The analogue of the Grand Unification Theorem for non-commutative architectures would express alignment conditions in terms of cyclic cohomology rather than Hochschild cohomology.

Open Problem 2: Prove a non-commutative Grand Unification Theorem in which the Hochschild obstruction is replaced by a cyclic cohomology class, and identify the classes of architectures (recurrent networks, graph transformers) for which the non-commutative theory is required.

9.6.3 Quantisation and Discrete Neural Networks

The entire framework developed in this monograph assumes *continuous* parameter spaces and *smooth* functions. Real-world deployment of neural networks involves *quantisation*: the replacement of continuous floating-point parameters with discrete integer or fixed-point representations. The quantisation map $q : \Theta \to \Theta_{\mathrm{disc}}$ is a morphism in **LinCoh** only if Θ_{disc} carries a diffeological structure — which it does (the discrete diffeology), but the quantisation map is not smooth.

Open Problem 3: Develop a theory of *quantised* linear cohesive morphisms that models the quantisation map as a geometric morphism between toposes (analogous to the transfer learning morphism of Chapter 7, Proposition 7.7), and prove a quantised version of the Grand Unification Theorem in which the smoothness conditions are replaced by Lipschitz or bounded-variation conditions.

9.6.4 The Alignment Completeness Problem

The Grand Unification Theorem gives a necessary and sufficient condition for a model to satisfy a specification: the specification must be globally valid in **LinCoh**. But it does not

address the *completeness problem* for alignment: is every true property of a model provable in $\mathcal{L}_{\mathrm{LC}}$?

Open Problem 4: Determine whether the linear cohesive type theory $\mathcal{L}_{\mathrm{LC}}$ is *complete* for neural computation in the following sense: for every property Φ that holds for all well-trained models of a given architecture (in a suitable statistical sense), is Φ provable in $\mathcal{L}_{\mathrm{LC}}$ from the architecture's axioms?

This is the neural-computation analogue of Gödel's completeness theorem for first-order logic — and, by Gödel's incompleteness theorem, we expect the answer to be negative for sufficiently expressive architectures. Making this precise — identifying which properties are Gödel-unprovable in $\mathcal{L}_{\mathrm{LC}}$ — would give a rigorous account of the fundamental limits of formal alignment.

9.7 Chapter Summary and Monograph Conclusion

9.7.1 Chapter Summary

This chapter defined the linear cohesive topos **LinCoh** as the unified mathematical universe for neural computation, proved its existence and basic properties, and established the Grand Unification Theorem (Theorem 9.4), which identifies six equivalent characterisations of universal specification satisfaction:

(I) Global validity in the internal language of **LinCoh**;

(II) Provability in $\mathcal{L}_{\mathrm{LC}}$;

(III) Vanishing of the sheaf cohomology obstruction $[o_\Phi] \in H^1(\mathcal{D}, \mathcal{R}_f)$;

(IV) Vanishing of the Hochschild class $[h_\Phi] \in \mathrm{HH}^2(A_f, M_{\mathcal{L}})$;

(V) The characteristic morphism χ_Φ factors through $\flat\Omega$;

(VI) The persistent homology barcode of the validity region contains no finite bars.

The quantitative measure of failure — the degree to which a specification is not satisfied — is the same in all six characterisations, up to the HKR isomorphism and the persistence stability theorem.

9.7.2 The Monograph in Retrospect

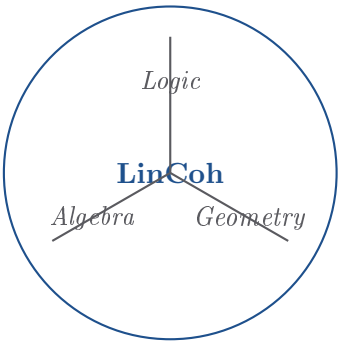
This monograph has developed three parallel journeys, which the Grand Unification Theorem shows to be a single journey seen from three different vantage points.

The **logical journey** (Chapters 1–3) began with the observation that a neural network layer is a morphism in a Cartesian closed category, identified the logic of resource consumption with linear logic, and showed that the forward-backward duality of learning is a lens in the Para construction. The destination: the category of learners **Learn** as a full subcategory of **LinCoh**.

The **homological journey** (Chapters 4–6) began with the observation that the loss landscape has non-trivial topology, developed the Morse, Hochschild, and sheaf-cohomological obstructions to training and generalisation, and applied the machinery to transformers, grokking, and domain generalisation. The destination: the obstruction classes $[o_\Phi] \in H^1$ and $[h_\Phi] \in \mathrm{HH}^2$ as computable invariants of any architecture.

The **geometric journey** (Chapters 7–8) began with the observation that the sheaves of the homological journey form a topos, developed the internal language of this topos as the correct framework for specifications, and added the cohesive structure of **Diff** to handle smoothness and differential geometry. The destination: the Coherence Theorem and the Cohesive Coherence Theorem as special cases of the Grand Unification Theorem.

The three destinations are the same place, viewed from logic, algebra, and geometry respectively. This place is the linear cohesive topos.



The open problems of Section 9.6 point outward from this place: toward ∞-toposes, noncommutative geometry, quantisation, and the fundamental limits of formal alignment. Each is a direction in which the unified framework can grow. The framework itself is not a closed system but a foundation — the right foundation, we believe, for a rigorous mathematics of intelligence.

Chapter A

Background in Category Theory

A.1 Categories, Functors, Natural Transformations

A **category** $\mathcal{C}$ consists of objects, morphisms with source and target maps, identity morphisms $\mathrm{id}_A : A \to A$, and a composition $g \circ f$ satisfying unit laws and associativity. A **functor** $F : \mathcal{C} \to \mathcal{D}$ preserves composition and identities. A **natural transformation** $\eta : F \Rightarrow G$ assigns to each A a morphism $\eta_A : F(A) \to G(A)$ commuting with all morphisms.

A.2 Adjunctions, Limits, Colimits

$F \dashv G$ (**adjunction**) means $\mathcal{D}(F(A), B) \cong \mathcal{C}(A, G(B))$ naturally. Limits (terminal object, products, pullbacks, equalisers) and colimits (initial object, coproducts, pushouts) are universal constructions characterised by unique mediating morphisms.

A.3 Monoidal and $*$-Autonomous Categories

A **monoidal category** $(\mathcal{C}, \otimes, \mathbf{1})$ has an associative tensor bifunctor with unit $\mathbf{1}$, coherent up to natural isomorphisms. A **Cartesian closed category** (CCC) has finite products and exponential objects B^A with $\mathrm{ev} : B^A \times A \to B$, satisfying the universal property of currying.

A **$*$-autonomous category** is a symmetric monoidal category with a dualising object $\bot$ such that the double-dualisation map $A \to (A \multimap \bot) \multimap \bot$ is a natural isomorphism. The prototypical example relevant here is $\mathbf{FVect}_{\mathbb{R}}$ with $\bot = \mathbb{R}$.

A.4 The Para Construction and Lens Categories

For a symmetric monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$, $\mathbf{Para}(\mathcal{C})$ has the same objects but morphisms from A to B are pairs (Θ, f) with $f : \Theta \otimes A \to B$, composed by tensoring parameter objects. The **lens category** $\mathbf{Lens}(\mathcal{C})$ over a Cartesian $\mathcal{C}$ has objects pairs (S, A) and morphisms $(\mathsf{get} : S \to A, \mathsf{put} : S \times B \to T)$.

Chapter B

Background in Algebraic Topology

B.1 Singular Homology and Cohomology

The **singular chain complex** $C_\bullet(X;\mathbb{Z})$ of a space X has $C_n = \mathbb{Z}[\mathrm{Sing}_n(X)]$ with boundary maps $\partial_n = \sum_{k=0}^{n}(-1)^k d_k$. The **homology groups** $H_n(X;\mathbb{Z}) = \ker\partial_n / \operatorname{im}\partial_{n+1}$ are homotopy-invariant. Dually, cohomology $H^n(X;\mathbb{Z}) = \ker d^n / \operatorname{im} d^{n-1}$ carries a cup product making $H^*(X;\mathbb{Z})$ a graded ring.

B.2 de Rham Cohomology

For a smooth manifold M, the **de Rham complex** $\Omega^\bullet(M)$ has differential $d : \Omega^n \to \Omega^{n+1}$ with $d^2 = 0$. By the de Rham theorem, $H^n_{\mathrm{dR}}(M) \cong H^n(M;\mathbb{R})$ (singular cohomology with real coefficients).

B.3 Morse Theory

A **Morse function** $f : M \to \mathbb{R}$ on a compact manifold has only non-degenerate critical points (all eigenvalues of the Hessian non-zero). The **Morse complex** has generators at critical points and differentials counting gradient flow lines. Morse homology equals singular homology. The **Morse inequalities** $c_k \geq \beta_k$ relate critical point counts c_k (by index) to Betti numbers β_k.

B.4 Persistent Homology

A **persistence module** $\mathbb{V} : (\mathbb{R}, \leq) \to \mathbf{Ab}$ is a family of abelian groups with order-compatible maps. Tame modules decompose into **interval modules** $\mathbf{I}[b,d)$; the multiset $\{[b_i, d_i)\}$ is the **barcode**. The **stability theorem**: $d_{\mathrm{bottle}}(\mathrm{dgm}(\mathbb{V}), \mathrm{dgm}(\mathbb{V}')) \leq \|f - f'\|_\infty$.

Chapter C

Background in Logic and Type Theory

C.1 Natural Deduction and Sequent Calculus

Intuitionistic natural deduction derives propositions from hypotheses via introduction and elimination rules for each connective, without the law of excluded middle. The **sequent calculus** derives sequents $\Gamma \vdash A$ with left/right rules; **cut elimination** (Gentzen's Hauptsatz) shows every proof has a cut-free normal form.

C.2 The Simply Typed Lambda Calculus

Types: $\tau ::= b \mid \tau_1 \to \tau_2$. **Terms**: variables $x : \tau$, abstractions $\lambda x{:}\tau.\, t$, applications $f\, a$. **Reduction**: $(\lambda x.\, t)\, a \to_\beta t[a/x]$; strong normalisation holds by Tait reducibility. The CHL correspondence bijects proofs of $P \Rightarrow Q$ with terms of type $\tau_P \to \tau_Q$.

C.3 Martin-Löf Type Theory (MLTT)

Dependent product $\prod_{x:A} B(x)$ classifies functions f with $f(a) : B(a)$ for all $a : A$ (corresponds to $\forall$). **Dependent sum** $\sum_{x:A} B(x)$ classifies pairs (a, b) with $b : B(a)$ (corresponds to $\exists$). MLTT is the internal language of locally Cartesian closed categories and of elementary toposes.

C.4 Linear Logic (Girard 1987)

Linear logic removes weakening and contraction, making logic resource-sensitive. **Multiplicative connectives**: $A \otimes B$ (both resources), $A \parr B$ (unified resource), $A \multimap B = A^\perp \parr B$ (linear implication). **Additive connectives**: $A \,\&\, B$ (choose one), $A \oplus B$ (one given). **Exponentials**: $!A$ (reusable resource, licensing contraction and weakening for A). The **Girard translation** $(-)^\circ$ embeds intuitionistic logic into linear logic: $(A \Rightarrow B)^\circ = !A^\circ \multimap B^\circ$.

Chapter D

Notation and Symbol Tables

D.1 Categorical Notation

Symbol	Meaning
$\mathbf{C}, \mathcal{E}, \mathcal{H}$	Generic categories or toposes
$\mathbf{Set}, \mathbf{Ab}, \mathbf{Vect}$	Sets, abelian groups, real vector spaces
$\mathbf{Smooth}, \mathbf{Diff}$	Smooth manifolds / diffeological spaces
$\mathbf{Para}(\mathcal{C})$	Para construction on $\mathcal{C}$
$\mathbf{Lens}(\mathcal{C})$	Lens category over $\mathcal{C}$
$\mathbf{Learn}$	Category of learners
$\mathbf{LinCoh}$	Linear cohesive topos
$\mathrm{Sh}\, X$	Sheaves on X; $\mathrm{PSh}\,\mathcal{C}$ = presheaves
$F \dashv G$	F left adjoint to G
$\mathbf{1}, \bot$	Monoidal unit, dualising object

D.2 Linear Logic and Type Theory

Symbol	Meaning
$A \otimes B$	Multiplicative conjunction
$A \parr B$	Multiplicative disjunction (par)
$A \multimap B$	Linear implication ($= A^{\bot} \parr B$)
$!A$	Of-course / exponential modality
$[\![\phi]\!]$	Semantic interpretation of formula ϕ
$U \Vdash \phi$	Kripke–Joyal forcing of ϕ at U
$\mathbf{\Pi}A$, $\flat A$, $\sharp A$	Shape, flat, sharp cohesive modalities

D.3 Homological and Geometric Notation

Symbol	Meaning
$H_n(X;\mathbb{Z})$, $H^n(X;\mathbb{Z})$	Singular homology / cohomology
$H^n_{\mathrm{dR}}(X)$	de Rham cohomology
$\mathrm{HH}^n(A, M)$	Hochschild cohomology of algebra A
$H^n(X, \mathcal{F})$	Sheaf cohomology
$\beta_k = \operatorname{rank} H_k(X;\mathbb{Z})$	Betti numbers
$\chi(X) = \sum (-1)^k \beta_k$	Euler characteristic
c_k	Number of Morse critical points of index k
$[b, d)$	Persistence interval (bar)
$\Omega_{\mathcal{E}}$	Subobject classifier of topos $\mathcal{E}$

Bibliography

The references are organised thematically. Within each section they follow chronological order of first publication.

I. Curry–Howard–Lambek Correspondence and Type Theory

Curry, H.B. (1934). Functionality in combinatory logic. *Proceedings of the National Academy of Sciences*, 20(11), 584–590.

Howard, W.A. (1980). The formulae-as-types notion of construction. In Seldin & Hindley (eds.), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, pp. 479–490.

Lambek, J. (1969). Deductive systems and categories II. *Lecture Notes in Mathematics*, 86, 76–122.

Lambek, J. & Scott, P.J. (1986). *Introduction to Higher Order Categorical Logic.* Cambridge Studies in Advanced Mathematics, vol. 7. Cambridge University Press.

Martin-Löf, P. (1975). An intuitionistic theory of types: Predicative part. In *Logic Colloquium '73*, North-Holland, pp. 73–118.

Girard, J.-Y., Lafont, Y. & Taylor, P. (1989). *Proofs and Types.* Cambridge Tracts in Theoretical Computer Science, vol. 7. Cambridge University Press.

II. Linear Logic and Proof Nets

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science*, 50(1), 1–101.

Danos, V. & Regnier, L. (1989). The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3), 181–203.

Seely, R.A.G. (1989). Linear logic, $*$-autonomous categories and cofree coalgebras. In *Categories in Computer Science and Logic*, AMS, pp. 371–382.

III. Lenses, Optics, and the Para Construction

Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C. & Schmitt, A. (2007). Combinators for bidirectional tree transformations. *ACM TOPLAS*, 29(3), article 17.

Fong, B., Spivak, D. & Tuyéras, R. (2019). Backprop as functor: A compositional perspective on supervised learning. In *Proceedings of LICS 2019*, IEEE.

Riley, M. (2018). *Categories of Optics.* arXiv:1809.00738.

Cruttwell, G.S.H., Gavranović, B., Ghani, N., Wilson, P. & Zanasi, F. (2022). Categorical foundations of gradient-based learning. In *Proceedings of ESOP 2022*, Springer LNCS 13240, pp. 1–28.

Gavranović, B. (2022). *Compositional Deep Learning.* PhD thesis, University of Strathclyde.

IV. Homological Algebra

Hochschild, G. (1945). On the cohomology groups of an associative algebra. *Annals of Mathematics*, 46(1), 58–67.

Cartan, H. & Eilenberg, S. (1956). *Homological Algebra.* Princeton University Press.

Weibel, C.A. (1994). *An Introduction to Homological Algebra.* Cambridge Studies in Advanced Mathematics, vol. 38. Cambridge University Press.

Loday, J.-L. (1998). *Cyclic Homology*, 2nd ed. Grundlehren der mathematischen Wissenschaften, vol. 301. Springer.

V. Morse Theory and Topology of Neural Networks

Milnor, J. (1963). *Morse Theory.* Annals of Mathematics Studies, vol. 51. Princeton University Press.

Edelsbrunner, H., Letscher, D. & Zomorodian, A. (2002). Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4), 511–533.

Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S. & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 27.

Chazal, F. & Michel, B. (2021). An introduction to topological data analysis. *Frontiers in Artificial Intelligence*, 4, 667963.

VI. Sheaf Theory and Domain Generalisation

Grothendieck, A. (1957). Sur quelques points d'algèbre homologique. *Tôhoku Mathematical Journal*, 9(2), 119–221.

Hansen, J. & Ghrist, R. (2020). Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3, 315–358.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F. & Vaughan, J.W. (2010). A theory of learning from different domains. *Machine Learning*, 79(1–2), 151–175.

Arjovsky, M., Bottou, L., Gulrajani, I. & Lopez-Paz, D. (2019). *Invariant Risk Minimization.* arXiv:1907.02893.

VII. Topos Theory

Lawvere, F.W. (1969). Adjointness in foundations. *Dialectica*, 23(3–4), 281–296.

Tierney, M. (1972). Sheaf theory and the continuum hypothesis. In *Toposes, Algebraic Geometry and Logic*, Springer LNM 274, pp. 13–42.

Joyal, A. & Tierney, M. (1984). *An Extension of the Galois Theory of Grothendieck.* Memoirs of the AMS, vol. 309.

Mac Lane, S. & Moerdijk, I. (1994). *Sheaves in Geometry and Logic: A First Introduction to Topos Theory.* Universitext. Springer.

Johnstone, P.T. (2002). *Sketches of an Elephant: A Topos Theory Compendium*, vols. 1–2. Oxford Logic Guides. Oxford University Press.

VIII. Cohesive Toposes and Differential Cohomology

Lawvere, F.W. (2007). Axiomatic cohesion. *Theory and Applications of Categories*, 19(3), 41–49.

Baez, J.C. & Hoffnung, A.E. (2011). Convenient categories of smooth spaces. *Transactions of the AMS*, 363(11), 5789–5825.

Kriegl, A. & Michor, P.W. (1997). *The Convenient Setting of Global Analysis.* Mathematical Surveys and Monographs, vol. 53. American Mathematical Society.

Schreiber, U. (2013). *Differential Cohomology in a Cohesive Infinity-Topos.* arXiv:1310.7930.

IX. Transformers, Scaling Laws, and Empirical Phenomena

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. In *NeurIPS 2017*.

Power, A., Burda, Y., Edwards, H., Goodfellow, I. & Misra, V. (2022). Grokking: Generalisation beyond overfitting on small algorithmic datasets. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. & Amodei, D. (2020). *Scaling Laws for Neural Language Models.* arXiv:2001.08361.

Hoffmann, J., Borgeaud, S., Mensch, A. et al. (2022). Training compute-optimal large language models. In *NeurIPS 2022*.

Wei, J., Tay, Y., Bommasani, R. et al. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Michel, H., Levy, O. & Neubig, G. (2019). Are sixteen heads really better than one? In *NeurIPS 2019*.

Nanda, N., Chan, L., Lieberum, T., Smith, J. & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. In *ICLR 2023*.

X. Information Geometry and Natural Gradient

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2), 251–276.

Amari, S. & Nagaoka, H. (2000). *Methods of Information Geometry.* Translations of Mathematical Monographs, vol. 191. American Mathematical Society.